MakerDAO

Auto-generated by code2pdf

February 10, 2023

Contents

References index

C

C

D

D

I

I

I

L

L

L

L

M

M

O

P

R

R

R

S

T

T

T

T

T

V

V

W

W

W

Chapter 1

Core

1.1 contract Dai

```
// FIXME: This contract was altered compared to the production version.
// It doesn't use LibNote anymore.
// New deployments of this contract will need to include custom events (TO DO).

contract Dai {
    // --- Auth ---
    mapping (address => uint) public wards;

    // --- ERC20 Data ---
    string  public constant name     = "Dai Stablecoin";
    string  public constant symbol   = "DAI";
    string  public constant version  = "1";
    uint8   public constant decimals = 18;
    uint256 public totalSupply;

    mapping (address => uint)                      public balanceOf;
    mapping (address => mapping (address => uint)) public allowance;
    mapping (address => uint)                      public nonces;

    event Approval(address indexed src, address indexed guy, uint wad);
    event Transfer(address indexed src, address indexed dst, uint wad);

    // --- EIP712 niceties ---
    bytes32 public DOMAIN_SEPARATOR;
    // bytes32 public constant PERMIT_TYPEHASH = keccak256("Permit(address
        ↪ holder,address spender,uint256 nonce,uint256 expiry,bool allowed)");
    bytes32 public constant PERMIT_TYPEHASH = 0
        ↪ xea2aa0a1be11a07ed86d755c93467f4f82362b452371d1ba94d1715123511acb;
}
```

1.1.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "Dai/not-authorized");
        _;
    }
```

1.1.2 rely(guy) X a

```
    function rely(address guy) external auth { wards[guy] = 1; }
```

1.1.3 deny(guy) X a

```
    function deny(address guy) external auth { wards[guy] = 0; }
```

1.1.4 add(x, y)

```
    // --- Math ---
    function add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x);
    }
```

1.1.5 sub(x, y)

```
    function sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x);
    }
```

1.1.6 constructor(chainId_) X

```
    constructor(uint256 chainId_) public {
        wards[msg.sender] = 1;
        DOMAIN_SEPARATOR = keccak256(abi.encode(
            keccak256("EIP712Domain(string name,string version,uint256 chainId,
                ↪ address verifyingContract)"),
            keccak256(bytes(name)),
            keccak256(bytes(version)),
            chainId_,
            address(this)
        ));
    }
```

1.1.7 transfer(dst, wad) X

```
    // --- Token ---
    function transfer(address dst, uint wad) external returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

1.1.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint wad)
        public returns (bool)
    {
        require(balanceOf[src] >= wad, "Dai/insufficient-balance");
        if (src != msg.sender && allowance[src][msg.sender] != uint(-1)) {
            require(allowance[src][msg.sender] >= wad, "Dai/insufficient-
                ↪ allowance");
            allowance[src][msg.sender] = sub(allowance[src][msg.sender], wad);
        }
        balanceOf[src] = sub(balanceOf[src], wad);
        balanceOf[dst] = add(balanceOf[dst], wad);
        emit Transfer(src, dst, wad);
        return true;
    }
```

1.1.9 mint(usr, wad) X a

```
    function mint(address usr, uint wad) external auth {
        balanceOf[usr] = add(balanceOf[usr], wad);
        totalSupply    = add(totalSupply, wad);
        emit Transfer(address(0), usr, wad);
    }
```

1.1.10 burn(usr, wad) X

```
    function burn(address usr, uint wad) external {
        require(balanceOf[usr] >= wad, "Dai/insufficient-balance");
        if (usr != msg.sender && allowance[usr][msg.sender] != uint(-1)) {
            require(allowance[usr][msg.sender] >= wad, "Dai/insufficient-
                ↪ allowance");
            allowance[usr][msg.sender] = sub(allowance[usr][msg.sender], wad);
        }
        balanceOf[usr] = sub(balanceOf[usr], wad);
        totalSupply    = sub(totalSupply, wad);
        emit Transfer(usr, address(0), wad);
    }
```

1.1.11 approve(usr, wad) X

```
    function approve(address usr, uint wad) external returns (bool) {
        allowance[msg.sender][usr] = wad;
        emit Approval(msg.sender, usr, wad);
        return true;
    }
```

1.1.12 push(usr, wad) X

```
    // --- Alias ---
    function push(address usr, uint wad) external {
        transferFrom(msg.sender, usr, wad);
    }
```

1.1.13 pull(usr, wad) X

```
    function pull(address usr, uint wad) external {
        transferFrom(usr, msg.sender, wad);
    }
```

1.1.14 move(src, dst, wad) X

```
    function move(address src, address dst, uint wad) external {
        transferFrom(src, dst, wad);
    }
```

1.1.15 permit(holder, spender, nonce, expiry, allowed, v, r, s) X

```
    // --- Approve by signature ---
    function permit(address holder, address spender, uint256 nonce, uint256
       ↪ expiry,
                    bool allowed, uint8 v, bytes32 r, bytes32 s) external
    {
        bytes32 digest =
            keccak256(abi.encodePacked(
                "\x19\x01",
                DOMAIN_SEPARATOR,
                keccak256(abi.encode(PERMIT_TYPEHASH,
                                     holder,
                                     spender,
                                     nonce,
                                     expiry,
                                     allowed))
        ));

        require(holder != address(0), "Dai/invalid-address-0");
        require(holder == ecrecover(digest, v, r, s), "Dai/invalid-permit");
        require(expiry == 0 || now <= expiry, "Dai/permit-expired");
        require(nonce == nonces[holder]++, "Dai/invalid-nonce");
        uint wad = allowed ? uint(-1) : 0;
        allowance[holder][spender] = wad;
        emit Approval(holder, spender, wad);
    }
```

1.2 contract Spotter

```
contract Spotter {
    // --- Auth ---
    mapping (address => uint) public wards;

    mapping (bytes32 => Ilk) public ilks;

    VatLike public vat;  // CDP Engine
    uint256 public par;  // ref per dai [ray]

    uint256 public live;

    // --- Events ---
    event Poke(
      bytes32 ilk,
      bytes32 val,  // [wad]
      uint256 spot  // [ray]
    );

    // --- Math ---
    uint constant ONE = 10 ** 27;
}
```

1.2.1 struct Spotter.Ilk

```
    // --- Data ---
    struct Ilk {
        PipLike pip;  // Price Feed
        uint256 mat;  // Liquidation ratio [ray]
    }
```

1.2.2 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "Spotter/not-authorized");
        _;
    }
```

1.2.3 rely(guy) X a

```
    function rely(address guy) external auth { wards[guy] = 1;  }
```

1.2.4 deny(guy) X a

```
    function deny(address guy) external auth { wards[guy] = 0; }
```

1.2.5 constructor(vat_) X

```
    // --- Init ---
    constructor(address vat_) public {
        wards[msg.sender] = 1;
        vat = VatLike(vat_);
        par = ONE;
        live = 1;
    }
```

1.2.6 mul(x, y)

```
    function mul(uint x, uint y) internal pure returns (uint z) {
        require(y == 0 || (z = x * y) / y == x);
    }
```

1.2.7 rdiv(x, y)

```
    function rdiv(uint x, uint y) internal pure returns (uint z) {
        z = mul(x, ONE) / y;
    }
```

1.2.8 file(ilk, what, pip_) X a

```
    // --- Administration ---
    function file(bytes32 ilk, bytes32 what, address pip_) external auth {
        require(live == 1, "Spotter/not-live");
        if (what == "pip") ilks[ilk].pip = PipLike(pip_);
        else revert("Spotter/file-unrecognized-param");
    }
```

1.2.9 file(what, data) X a

```
    function file(bytes32 what, uint data) external auth {
        require(live == 1, "Spotter/not-live");
        if (what == "par") par = data;
        else revert("Spotter/file-unrecognized-param");
    }
```

1.2.10 file(ilk, what, data) X a

```
    function file(bytes32 ilk, bytes32 what, uint data) external auth {
        require(live == 1, "Spotter/not-live");
        if (what == "mat") ilks[ilk].mat = data;
        else revert("Spotter/file-unrecognized-param");
    }
```

1.2.11 poke(ilk) X

```
    // --- Update value ---
    function poke(bytes32 ilk) external {
        (bytes32 val, bool has) = ilks[ilk].pip.peek();
        uint256 spot = has ? rdiv(rdiv(mul(uint(val), 10 ** 9), par), ilks[ilk].
            ↪ mat) : 0;
        vat.file(ilk, "spot", spot);
        emit Poke(ilk, val, spot);
    }
```

1.2.12 cage() X a

```
    function cage() external auth {
        live = 0;
    }
```

1.3 contract Vat

```
// FIXME: This contract was altered compared to the production version.
// It doesn't use LibNote anymore.
// New deployments of this contract will need to include custom events (TO DO).

contract Vat {
    // --- Auth ---
    mapping (address => uint) public wards;

    mapping(address => mapping (address => uint)) public can;

    mapping (bytes32 => Ilk)                        public ilks;
    mapping (bytes32 => mapping (address => Urn )) public urns;
    mapping (bytes32 => mapping (address => uint)) public gem;  // [wad]
    mapping (address => uint256)                    public dai;  // [rad]
    mapping (address => uint256)                    public sin;  // [rad]

    uint256 public debt;  // Total Dai Issued    [rad]
    uint256 public vice;  // Total Unbacked Dai  [rad]
    uint256 public Line;  // Total Debt Ceiling  [rad]
    uint256 public live;  // Active Flag
}
```

1.3.1 struct Vat.Ilk

```
    // --- Data ---
    struct Ilk {
        uint256 Art;   // Total Normalised Debt     [wad]
        uint256 rate;  // Accumulated Rates         [ray]
        uint256 spot;  // Price with Safety Margin  [ray]
        uint256 line;  // Debt Ceiling              [rad]
        uint256 dust;  // Urn Debt Floor            [rad]
    }
```

1.3.2 struct Vat.Urn

```
    struct Urn {
        uint256 ink;   // Locked Collateral  [wad]
        uint256 art;   // Normalised Debt    [wad]
    }
```

1.3.3 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "Vat/not-authorized");
        _;
    }
```

1.3.4 rely(usr) X a

```
    function rely(address usr) external auth { require(live == 1, "Vat/not-live"
       ↪ ); wards[usr] = 1; }
```

1.3.5 deny(usr) X a

```
    function deny(address usr) external auth { require(live == 1, "Vat/not-live"
       ↪ ); wards[usr] = 0; }
```

1.3.6 hope(usr) X

```
    function hope(address usr) external { can[msg.sender][usr] = 1; }
```

1.3.7 nope(usr) X

```
    function nope(address usr) external { can[msg.sender][usr] = 0; }
```

1.3.8 wish(bit, usr)

```
    function wish(address bit, address usr) internal view returns (bool) {
        return either(bit == usr, can[bit][usr] == 1);
    }
```

1.3.9 constructor() X

```
    // --- Init ---
    constructor() public {
        wards[msg.sender] = 1;
        live = 1;
    }
```

1.3.10 _add(x, y)

```
    // --- Math ---
    function _add(uint x, int y) internal pure returns (uint z) {
        z = x + uint(y);
        require(y >= 0 || z <= x);
        require(y <= 0 || z >= x);
    }
```

1.3.11 _sub(x, y)

```
    function _sub(uint x, int y) internal pure returns (uint z) {
        z = x - uint(y);
        require(y <= 0 || z <= x);
        require(y >= 0 || z >= x);
    }
```

1.3.12 _mul(x, y)

```
    function _mul(uint x, int y) internal pure returns (int z) {
        z = int(x) * y;
        require(int(x) >= 0);
        require(y == 0 || z / y == int(x));
    }
```

1.3.13 _add(x, y)

```
    function _add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x);
    }
```

1.3.14 _sub(x, y)

```
function _sub(uint x, uint y) internal pure returns (uint z) {
    require((z = x - y) <= x);
}
```

1.3.15 _mul(x, y)

```
function _mul(uint x, uint y) internal pure returns (uint z) {
    require(y == 0 || (z = x * y) / y == x);
}
```

1.3.16 init(ilk) X a

```
// --- Administration ---
function init(bytes32 ilk) external auth {
    require(ilks[ilk].rate == 0, "Vat/ilk-already-init");
    ilks[ilk].rate = 10 ** 27;
}
```

1.3.17 file(what, data) X a

```
function file(bytes32 what, uint data) external auth {
    require(live == 1, "Vat/not-live");
    if (what == "Line") Line = data;
    else revert("Vat/file-unrecognized-param");
}
```

1.3.18 file(ilk, what, data) X a

```
function file(bytes32 ilk, bytes32 what, uint data) external auth {
    require(live == 1, "Vat/not-live");
    if (what == "spot") ilks[ilk].spot = data;
    else if (what == "line") ilks[ilk].line = data;
    else if (what == "dust") ilks[ilk].dust = data;
    else revert("Vat/file-unrecognized-param");
}
```

1.3.19 cage() X a

```
function cage() external auth {
    live = 0;
}
```

1.3.20 slip(ilk, usr, wad) X a

```
// --- Fungibility ---
function slip(bytes32 ilk, address usr, int256 wad) external auth {
    gem[ilk][usr] = _add(gem[ilk][usr], wad);
}
```

1.3.21 flux(ilk, src, dst, wad) X

```
function flux(bytes32 ilk, address src, address dst, uint256 wad) external {
    require(wish(src, msg.sender), "Vat/not-allowed");
    gem[ilk][src] = _sub(gem[ilk][src], wad);
    gem[ilk][dst] = _add(gem[ilk][dst], wad);
}
```

1.3.22 move(src, dst, rad) X

```
    function move(address src, address dst, uint256 rad) external {
        require(wish(src, msg.sender), "Vat/not-allowed");
        dai[src] = _sub(dai[src], rad);
        dai[dst] = _add(dai[dst], rad);
    }
```

1.3.23 either(x, y)

```
    function either(bool x, bool y) internal pure returns (bool z) {
        assembly{ z := or(x, y)}
    }
```

1.3.24 both(x, y)

```
    function both(bool x, bool y) internal pure returns (bool z) {
        assembly{ z := and(x, y)}
    }
```

1.3.25 frob(i, u, v, w, dink, dart) X

```
    // --- CDP Manipulation ---
    function frob(bytes32 i, address u, address v, address w, int dink, int dart
       ↪ ) external {
        // system is live
        require(live == 1, "Vat/not-live");

        Urn memory urn = urns[i][u];
        Ilk memory ilk = ilks[i];
        // ilk has been initialised
        require(ilk.rate != 0, "Vat/ilk-not-init");

        urn.ink = _add(urn.ink, dink);
        urn.art = _add(urn.art, dart);
        ilk.Art = _add(ilk.Art, dart);

        int dtab = _mul(ilk.rate, dart);
        uint tab = _mul(ilk.rate, urn.art);
        debt     = _add(debt, dtab);

        // either debt has decreased, or debt ceilings are not exceeded
        require(either(dart <= 0, both(_mul(ilk.Art, ilk.rate) <= ilk.line, debt
            ↪  <= Line)), "Vat/ceiling-exceeded");
        // urn is either less risky than before, or it is safe
        require(either(both(dart <= 0, dink >= 0), tab <= _mul(urn.ink, ilk.spot
            ↪ )), "Vat/not-safe");

        // urn is either more safe, or the owner consents
        require(either(both(dart <= 0, dink >= 0), wish(u, msg.sender)), "Vat/
            ↪ not-allowed-u");
        // collateral src consents
        require(either(dink <= 0, wish(v, msg.sender)), "Vat/not-allowed-v");
        // debt dst consents
        require(either(dart >= 0, wish(w, msg.sender)), "Vat/not-allowed-w");

        // urn has no debt, or a non-dusty amount
        require(either(urn.art == 0, tab >= ilk.dust), "Vat/dust");

        gem[i][v] = _sub(gem[i][v], dink);
        dai[w]    = _add(dai[w],    dtab);

        urns[i][u] = urn;
        ilks[i]    = ilk;
    }
```

1.3.26 fork(ilk, src, dst, dink, dart) X

```
    // --- CDP Fungibility ---
    function fork(bytes32 ilk, address src, address dst, int dink, int dart)
       ↪ external {
        Urn storage u = urns[ilk][src];
        Urn storage v = urns[ilk][dst];
        Ilk storage i = ilks[ilk];

        u.ink = _sub(u.ink, dink);
        u.art = _sub(u.art, dart);
        v.ink = _add(v.ink, dink);
        v.art = _add(v.art, dart);

        uint utab = _mul(u.art, i.rate);
        uint vtab = _mul(v.art, i.rate);

        // both sides consent
        require(both(wish(src, msg.sender), wish(dst, msg.sender)), "Vat/not-
            ↪ allowed");

        // both sides safe
        require(utab <= _mul(u.ink, i.spot), "Vat/not-safe-src");
        require(vtab <= _mul(v.ink, i.spot), "Vat/not-safe-dst");

        // both sides non-dusty
        require(either(utab >= i.dust, u.art == 0), "Vat/dust-src");
        require(either(vtab >= i.dust, v.art == 0), "Vat/dust-dst");
    }
```

1.3.27 grab(i, u, v, w, dink, dart) X a

```
    // --- CDP Confiscation ---
    function grab(bytes32 i, address u, address v, address w, int dink, int dart
       ↪ ) external auth {
        Urn storage urn = urns[i][u];
        Ilk storage ilk = ilks[i];

        urn.ink = _add(urn.ink, dink);
        urn.art = _add(urn.art, dart);
        ilk.Art = _add(ilk.Art, dart);

        int dtab = _mul(ilk.rate, dart);

        gem[i][v] = _sub(gem[i][v], dink);
        sin[w]    = _sub(sin[w],    dtab);
        vice      = _sub(vice,      dtab);
    }
```

1.3.28 heal(rad) X

```
    // --- Settlement ---
    function heal(uint rad) external {
        address u = msg.sender;
        sin[u] = _sub(sin[u], rad);
        dai[u] = _sub(dai[u], rad);
        vice   = _sub(vice,   rad);
        debt   = _sub(debt,   rad);
    }
```

1.3.29 suck(u, v, rad) X a

```
    function suck(address u, address v, uint rad) external auth {
        sin[u] = _add(sin[u], rad);
        dai[v] = _add(dai[v], rad);
        vice   = _add(vice,   rad);
        debt   = _add(debt,   rad);
    }
```

1.3.30 fold(i, u, rate) X a

```
    // --- Rates ---
    function fold(bytes32 i, address u, int rate) external auth {
        require(live == 1, "Vat/not-live");
        Ilk storage ilk = ilks[i];
        ilk.rate = _add(ilk.rate, rate);
        int rad  = _mul(ilk.Art, rate);
        dai[u]   = _add(dai[u], rad);
        debt     = _add(debt,   rad);
    }
```

Chapter 2

Liquidations

2.1 contract LinearDecrease

```
contract LinearDecrease is Abacus {

    // --- Auth ---
    mapping (address => uint256) public wards;

    // --- Data ---
    uint256 public tau;  // Seconds after auction start when the price reaches
        ↪ zero [seconds]

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);

    event File(bytes32 indexed what, uint256 data);

    // --- Math ---
    uint256 constant RAY = 10 ** 27;
}
```

2.1.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "LinearDecrease/not-authorized");
        _;
    }
```

2.1.2 rely(usr) X a

```
    function rely(address usr) external auth { wards[usr] = 1; emit Rely(usr); }
```

2.1.3 deny(usr) X a

```
    function deny(address usr) external auth { wards[usr] = 0; emit Deny(usr); }
```

2.1.4 constructor() X

```
    // --- Init ---
    constructor() public {
        wards[msg.sender] = 1;
        emit Rely(msg.sender);
    }
```

2.1.5 file(what, data) X a

```
    // --- Administration ---
    function file(bytes32 what, uint256 data) external auth {
        if (what ==  "tau") tau = data;
        else revert("LinearDecrease/file-unrecognized-param");
        emit File(what, data);
    }
```

2.1.6 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x);
    }
```

2.1.7 mul(x, y)

```
    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require(y == 0 || (z = x * y) / y == x);
    }
```

2.1.8 rmul(x, y)

```
    function rmul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = x * y;
        require(y == 0 || z / y == x);
        z = z / RAY;
    }
```

2.1.9 price(top, dur)

```
    // Price calculation when price is decreased linearly in proportion to time:
    // tau: The number of seconds after the start of the auction where the price
        ↪  will hit 0
    // top: Initial price
    // dur: current seconds since the start of the auction
    //
    // Returns y = top * ((tau - dur) / tau)
    //
    // Note the internal call to mul multiples by RAY, thereby ensuring that the
        ↪  rmul calculation
    // which utilizes top and tau (RAY values) is also a RAY value.
    function price(uint256 top, uint256 dur) override external view returns (
        ↪ uint256) {
        if (dur >= tau) return 0;
        return rmul(top, mul(tau - dur, RAY) / tau);
    }
```

2.2 contract StairstepExponentialDecrease

```
contract StairstepExponentialDecrease is Abacus {

    // --- Auth ---
    mapping (address => uint256) public wards;

    // --- Data ---
    uint256 public step; // Length of time between price drops [seconds]
    uint256 public cut;  // Per-step multiplicative factor     [ray]

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);

    event File(bytes32 indexed what, uint256 data);

    // --- Math ---
    uint256 constant RAY = 10 ** 27;
}
```

2.2.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "StairstepExponentialDecrease/not-
            ↪ authorized");
        _;
    }
```

2.2.2 rely(usr) X a

```
    function rely(address usr) external auth { wards[usr] = 1; emit Rely(usr); }
```

2.2.3 deny(usr) X a

```
    function deny(address usr) external auth { wards[usr] = 0; emit Deny(usr); }
```

2.2.4 constructor() X

```
    // --- Init ---
    // @notice: `cut` and `step` values must be correctly set for
    //     this contract to return a valid price
    constructor() public {
        wards[msg.sender] = 1;
        emit Rely(msg.sender);
    }
```

2.2.5 file(what, data) X a

```
    // --- Administration ---
    function file(bytes32 what, uint256 data) external auth {
        if      (what ==  "cut") require((cut = data) <= RAY, "
            ↪ StairstepExponentialDecrease/cut-gt-RAY");
        else if (what == "step") step = data;
        else revert("StairstepExponentialDecrease/file-unrecognized-param");
        emit File(what, data);
    }
```

2.2.6 rmul(x, y)

```
function rmul(uint256 x, uint256 y) internal pure returns (uint256 z) {
    z = x * y;
    require(y == 0 || z / y == x);
    z = z / RAY;
}
```

2.2.7 rpow(x, n, b)

```
// optimized version from dss PR #78
function rpow(uint256 x, uint256 n, uint256 b) internal pure returns (
   ↪ uint256 z) {
    assembly {
        switch n case 0 { z := b }
        default {
            switch x case 0 { z := 0 }
            default {
                switch mod(n, 2) case 0 { z := b } default { z := x }
                let half := div(b, 2)  // for rounding.
                for { n := div(n, 2) } n { n := div(n,2) } {
                    let xx := mul(x, x)
                    if shr(128, x) { revert(0,0) }
                    let xxRound := add(xx, half)
                    if lt(xxRound, xx) { revert(0,0) }
                    x := div(xxRound, b)
                    if mod(n,2) {
                        let zx := mul(z, x)
                        if and(iszero(iszero(x)), iszero(eq(div(zx, x), z)))
                           ↪  { revert(0,0) }
                        let zxRound := add(zx, half)
                        if lt(zxRound, zx) { revert(0,0) }
                        z := div(zxRound, b)
                    }
                }
            }
        }
    }
}
```

2.2.8 price(top, dur)

```
// top: initial price
// dur: seconds since the auction has started
// step: seconds between a price drop
// cut: cut encodes the percentage to decrease per step.
//   For efficiency, the values is set as (1 - (% value / 100)) * RAY
//   So, for a 1% decrease per step, cut would be (1 - 0.01) * RAY
//
// returns: top * (cut ^ dur)
//
//
function price(uint256 top, uint256 dur) override external view returns (
   ↪ uint256) {
    return rmul(top, rpow(cut, dur / step, RAY));
}
```

2.3 contract ExponentialDecrease

```
// While an equivalent function can be obtained by setting step = 1 in
    ↪ StairstepExponentialDecrease,
// this continous (i.e. per-second) exponential decrease has be implemented as
    ↪ it is more gas-efficient
// than using the stairstep version with step = 1 (primarily due to 1 fewer
    ↪ SLOAD per price calculation).
contract ExponentialDecrease is Abacus {

    // --- Auth ---
    mapping (address => uint256) public wards;

    // --- Data ---
    uint256 public cut;  // Per-second multiplicative factor [ray]

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);

    event File(bytes32 indexed what, uint256 data);

    // --- Math ---
    uint256 constant RAY = 10 ** 27;
}
```

2.3.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "ExponentialDecrease/not-authorized");
        _;
    }
```

2.3.2 rely(usr) X a

```
    function rely(address usr) external auth { wards[usr] = 1; emit Rely(usr); }
```

2.3.3 deny(usr) X a

```
    function deny(address usr) external auth { wards[usr] = 0; emit Deny(usr); }
```

2.3.4 constructor() X

```
    // --- Init ---
    // @notice: `cut` value must be correctly set for
    //     this contract to return a valid price
    constructor() public {
        wards[msg.sender] = 1;
        emit Rely(msg.sender);
    }
```

2.3.5 file(what, data) X a

```
    // --- Administration ---
    function file(bytes32 what, uint256 data) external auth {
        if      (what ==  "cut") require((cut = data) <= RAY, "
            ↪ ExponentialDecrease/cut-gt-RAY");
        else revert("ExponentialDecrease/file-unrecognized-param");
        emit File(what, data);
    }
```

2.3.6 rmul(x, y)

```
function rmul(uint256 x, uint256 y) internal pure returns (uint256 z) {
    z = x * y;
    require(y == 0 || z / y == x);
    z = z / RAY;
}
```

2.3.7 rpow(x, n, b)

```
// optimized version from dss PR #78
function rpow(uint256 x, uint256 n, uint256 b) internal pure returns (
   ↪ uint256 z) {
    assembly {
        switch n case 0 { z := b }
        default {
            switch x case 0 { z := 0 }
            default {
                switch mod(n, 2) case 0 { z := b } default { z := x }
                let half := div(b, 2)  // for rounding.
                for { n := div(n, 2) } n { n := div(n,2) } {
                    let xx := mul(x, x)
                    if shr(128, x) { revert(0,0) }
                    let xxRound := add(xx, half)
                    if lt(xxRound, xx) { revert(0,0) }
                    x := div(xxRound, b)
                    if mod(n,2) {
                        let zx := mul(z, x)
                        if and(iszero(iszero(x)), iszero(eq(div(zx, x), z)))
                           ↪  { revert(0,0) }
                        let zxRound := add(zx, half)
                        if lt(zxRound, zx) { revert(0,0) }
                        z := div(zxRound, b)
                    }
                }
            }
        }
    }
}
```

2.3.8 price(top, dur)

```
// top: initial price
// dur: seconds since the auction has started
// cut: cut encodes the percentage to decrease per second.
//   For efficiency, the values is set as (1 - (% value / 100)) * RAY
//   So, for a 1% decrease per second, cut would be (1 - 0.01) * RAY
//
// returns: top * (cut ^ dur)
//
function price(uint256 top, uint256 dur) override external view returns (
   ↪ uint256) {
    return rmul(top, rpow(cut, dur, RAY));
}
```

2.4 contract Clipper

```
contract Clipper {
    // --- Auth ---
    mapping (address => uint256) public wards;

    // --- Data ---
    bytes32  immutable public ilk;   // Collateral type of this Clipper
    VatLike  immutable public vat;   // Core CDP Engine

    DogLike     public dog;      // Liquidation module
    address     public vow;      // Recipient of dai raised in auctions
    SpotterLike public spotter;  // Collateral price module
    AbacusLike  public calc;     // Current price calculator

    uint256 public buf;    // Multiplicative factor to increase starting price
        ↪                  [ray]
    uint256 public tail;   // Time elapsed before auction reset
        ↪                                 [seconds]
    uint256 public cusp;   // Percentage drop before auction reset
        ↪                              [ray]
    uint64  public chip;   // Percentage of tab to suck from vow to incentivize
        ↪ keepers         [wad]
    uint192 public tip;    // Flat fee to suck from vow to incentivize keepers
        ↪                  [rad]
    uint256 public chost;  // Cache the ilk dust times the ilk chop to prevent
        ↪ excessive SLOADs [rad]

    uint256   public kicks;   // Total auctions
    uint256[] public active;  // Array of active auction ids
    mapping(uint256 => Sale) public sales;

    uint256 internal locked;

    // Levels for circuit breaker
    // 0: no breaker
    // 1: no new kick()
    // 2: no new kick() or redo()
    // 3: no new kick(), redo(), or take()
    uint256 public stopped = 0;

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);

    event File(bytes32 indexed what, uint256 data);
    event File(bytes32 indexed what, address data);

    event Kick(
        uint256 indexed id,
        uint256 top,
        uint256 tab,
        uint256 lot,
        address indexed usr,
        address indexed kpr,
        uint256 coin
    );
    event Take(
        uint256 indexed id,
        uint256 max,
        uint256 price,
        uint256 owe,
        uint256 tab,
        uint256 lot,
        address indexed usr
    );
    event Redo(
        uint256 indexed id,
        uint256 top,
        uint256 tab,
        uint256 lot,
        address indexed usr,
        address indexed kpr,
        uint256 coin
    );

    event Yank(uint256 id);

    // --- Math ---
    uint256 constant BLN = 10 **  9;
    uint256 constant WAD = 10 ** 18;
    uint256 constant RAY = 10 ** 27;
}
```

2.4.1 struct Clipper.Sale

```
    struct Sale {
        uint256 pos;  // Index in active array
        uint256 tab;  // Dai to raise       [rad]
        uint256 lot;  // collateral to sell [wad]
        address usr;  // Liquidated CDP
        uint96  tic;  // Auction start time
        uint256 top;  // Starting price     [ray]
    }
```

2.4.2 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "Clipper/not-authorized");
        _;
    }
```

2.4.3 modifier lock()

```
    // --- Synchronization ---
    modifier lock {
        require(locked == 0, "Clipper/system-locked");
        locked = 1;
        _;
        locked = 0;
    }
```

2.4.4 modifier isStopped(level)

```
    modifier isStopped(uint256 level) {
        require(stopped < level, "Clipper/stopped-incorrect");
        _;
    }
```

2.4.5 rely(usr) X a

```
    function rely(address usr) external auth { wards[usr] = 1; emit Rely(usr); }
```

2.4.6 deny(usr) X a

```
    function deny(address usr) external auth { wards[usr] = 0; emit Deny(usr); }
```

2.4.7 constructor(vat_, spotter_, dog_, ilk_) X

```
// --- Init ---
constructor(address vat_, address spotter_, address dog_, bytes32 ilk_)
    ↪ public {
    vat     = VatLike(vat_);
    spotter = SpotterLike(spotter_);
    dog     = DogLike(dog_);
    ilk     = ilk_;
    buf     = RAY;
    wards[msg.sender] = 1;
    emit Rely(msg.sender);
}
```

2.4.8 file(what, data) X a

```
// --- Administration ---
function file(bytes32 what, uint256 data) external auth lock {
    if      (what == "buf")          buf = data;
    else if (what == "tail")        tail = data;           // Time elapsed
        ↪ before auction reset
    else if (what == "cusp")        cusp = data;           // Percentage drop
        ↪  before auction reset
    else if (what == "chip")        chip = uint64(data);   // Percentage of
        ↪ tab to incentivize (max: 2^64 - 1 => 18.xxx WAD = 18xx%)
    else if (what == "tip")          tip = uint192(data);  // Flat fee to
        ↪ incentivize keepers (max: 2^192 - 1 => 6.277T RAD)
    else if (what == "stopped") stopped = data;            // Set breaker (0,
        ↪  1, 2, or 3)
    else revert("Clipper/file-unrecognized-param");
    emit File(what, data);
}
```

2.4.9 file(what, data) X a

```
function file(bytes32 what, address data) external auth lock {
    if (what == "spotter") spotter = SpotterLike(data);
    else if (what == "dog")    dog = DogLike(data);
    else if (what == "vow")    vow = data;
    else if (what == "calc")  calc = AbacusLike(data);
    else revert("Clipper/file-unrecognized-param");
    emit File(what, data);
}
```

2.4.10 min(x, y)

```
function min(uint256 x, uint256 y) internal pure returns (uint256 z) {
    z = x <= y ? x : y;
}
```

2.4.11 add(x, y)

```
function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
    require((z = x + y) >= x);
}
```

2.4.12 sub(x, y)

```
function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
    require((z = x - y) <= x);
}
```

2.4.13 mul(x, y)

```
    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require(y == 0 || (z = x * y) / y == x);
    }
```

2.4.14 wmul(x, y)

```
    function wmul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = mul(x, y) / WAD;
    }
```

2.4.15 rmul(x, y)

```
    function rmul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = mul(x, y) / RAY;
    }
```

2.4.16 rdiv(x, y)

```
    function rdiv(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = mul(x, RAY) / y;
    }
```

2.4.17 getFeedPrice()

```
    // --- Auction ---

    // get the price directly from the OSM
    // Could get this from rmul(Vat.ilks(ilk).spot, Spotter.mat()) instead, but
    // if mat has changed since the last poke, the resulting value will be
    // incorrect.
    function getFeedPrice() internal returns (uint256 feedPrice) {
        (PipLike pip, ) = spotter.ilks(ilk);
        (bytes32 val, bool has) = pip.peek();
        require(has, "Clipper/invalid-price");
        feedPrice = rdiv(mul(uint256(val), BLN), spotter.par());
    }
```

2.4.18 kick(tab, lot, usr, kpr) X a

```
    // start an auction
    // note: trusts the caller to transfer collateral to the contract
    // The starting price `top` is obtained as follows:
    //
    //     top = val * buf / par
    //
    // Where `val` is the collateral's unitary value in USD, `buf` is a
    // multiplicative factor to increase the starting price, and `par` is a
    // reference per DAI.
    function kick(
        uint256 tab,  // Debt                   [rad]
        uint256 lot,  // Collateral             [wad]
        address usr,  // Address that will receive any leftover collateral
        address kpr   // Address that will receive incentives
    ) external auth lock isStopped(1) returns (uint256 id) {
        // Input validation
        require(tab  >          0, "Clipper/zero-tab");
        require(lot  >          0, "Clipper/zero-lot");
        require(usr != address(0), "Clipper/zero-usr");
        id = ++kicks;
        require(id   >          0, "Clipper/overflow");

        active.push(id);

        sales[id].pos = active.length - 1;

        sales[id].tab = tab;
        sales[id].lot = lot;
        sales[id].usr = usr;
        sales[id].tic = uint96(block.timestamp);

        uint256 top;
        top = rmul(getFeedPrice(), buf);
        require(top > 0, "Clipper/zero-top-price");
        sales[id].top = top;

        // incentive to kick auction
        uint256 _tip  = tip;
        uint256 _chip = chip;
        uint256 coin;
        if (_tip > 0 || _chip > 0) {
            coin = add(_tip, wmul(tab, _chip));
            vat.suck(vow, kpr, coin);
        }

        emit Kick(id, top, tab, lot, usr, kpr, coin);
    }
```

2.4.19 redo(id, kpr) X

```
    // Reset an auction
    // See `kick` above for an explanation of the computation of `top`.
    function redo(
        uint256 id,  // id of the auction to reset
        address kpr  // Address that will receive incentives
    ) external lock isStopped(2) {
        // Read auction data
        address usr = sales[id].usr;
        uint96  tic = sales[id].tic;
        uint256 top = sales[id].top;

        require(usr != address(0), "Clipper/not-running-auction");

        // Check that auction needs reset
        // and compute current price [ray]
        (bool done,) = status(tic, top);
        require(done, "Clipper/cannot-reset");

        uint256 tab   = sales[id].tab;
        uint256 lot   = sales[id].lot;
        sales[id].tic = uint96(block.timestamp);

        uint256 feedPrice = getFeedPrice();
        top = rmul(feedPrice, buf);
        require(top > 0, "Clipper/zero-top-price");
        sales[id].top = top;

        // incentive to redo auction
        uint256 _tip  = tip;
        uint256 _chip = chip;
        uint256 coin;
        if (_tip > 0 || _chip > 0) {
            uint256 _chost = chost;
            if (tab >= _chost && mul(lot, feedPrice) >= _chost) {
                coin = add(_tip, wmul(tab, _chip));
                vat.suck(vow, kpr, coin);
            }
        }

        emit Redo(id, top, tab, lot, usr, kpr, coin);
    }
```

2.4.20 take(id, amt, max, who, data) X

```
    // Buy up to `amt` of collateral from the auction indexed by `id`.
    //
    // Auctions will not collect more DAI than their assigned DAI target,`tab`;
    // thus, if `amt` would cost more DAI than `tab` at the current price, the
    // amount of collateral purchased will instead be just enough to collect `
        ↪ tab` DAI.
    //
    // To avoid partial purchases resulting in very small leftover auctions that
        ↪  will
    // never be cleared, any partial purchase must leave at least `Clipper.chost
        ↪ `
    // remaining DAI target. `chost` is an asynchronously updated value equal to
    // (Vat.dust * Dog.chop(ilk) / WAD) where the values are understood to be
        ↪ determined
    // by whatever they were when Clipper.upchost() was last called. Purchase
        ↪ amounts
    // will be minimally decreased when necessary to respect this limit; i.e.,
        ↪ if the
    // specified `amt` would leave `tab < chost` but `tab > 0`, the amount
        ↪ actually
    // purchased will be such that `tab == chost`.
    //
    // If `tab <= chost`, partial purchases are no longer possible; that is, the
        ↪  remaining
    // collateral can only be purchased entirely, or not at all.
    function take(
        uint256 id,           // Auction id
        uint256 amt,          // Upper limit on amount of collateral to buy  [
            ↪ wad]
        uint256 max,          // Maximum acceptable price (DAI / collateral) [
            ↪ ray]
        address who,          // Receiver of collateral and external call
            ↪ address
        bytes calldata data   // Data to pass in external call; if length 0, no
            ↪ call is done
    ) external lock isStopped(3) {

        address usr = sales[id].usr;
        uint96  tic = sales[id].tic;

        require(usr != address(0), "Clipper/not-running-auction");

        uint256 price;
        {
            bool done;
            (done, price) = status(tic, sales[id].top);

            // Check that auction doesn't need reset
            require(!done, "Clipper/needs-reset");
        }

        // Ensure price is acceptable to buyer
        require(max >= price, "Clipper/too-expensive");

        uint256 lot = sales[id].lot;
        uint256 tab = sales[id].tab;
        uint256 owe;

        {
            // Purchase as much as possible, up to amt
            uint256 slice = min(lot, amt);  // slice <= lot

            // DAI needed to buy a slice of this sale
            owe = mul(slice, price);

            // Don't collect more than tab of DAI
            if (owe > tab) {
                // Total debt will be paid
                owe = tab;                      // owe' <= owe
                // Adjust slice
                slice = owe / price;            // slice' = owe' / price <= owe /
                    ↪ price == slice <= lot
            } else if (owe < tab && slice < lot) {
                // If slice == lot => auction completed => dust doesn't matter
                uint256 _chost = chost;
                if (tab - owe < _chost) {    // safe as owe < tab
                    // If tab <= chost, buyers have to take the entire lot.
                    require(tab > _chost, "Clipper/no-partial-purchase");
                    // Adjust amount to pay
                    owe = tab - _chost;      // owe' <= owe
                    // Adjust slice
                    slice = owe / price;     // slice' = owe' / price < owe /
                        ↪ price == slice < lot
                }
            }

            // Calculate remaining tab after operation
            tab = tab - owe;  // safe since owe <= tab
            // Calculate remaining lot after operation
            lot = lot - slice;

            // Send collateral to who
            vat.flux(ilk, address(this), who, slice);

            // Do external call (if data is defined) but to be
            // extremely careful we don't allow to do it to the two
            // contracts which the Clipper needs to be authorized
            DogLike dog_ = dog;
            if (data.length > 0 && who != address(vat) && who != address(dog_))
                ↪ {
                ClipperCallee(who).clipperCall(msg.sender, owe, slice, data);
            }

            // Get DAI from caller
            vat.move(msg.sender, vow, owe);

            // Removes Dai out for liquidation from accumulator
            dog_.digs(ilk, lot == 0 ? tab + owe : owe);
        }

        if (lot == 0) {
            _remove(id);
        } else if (tab == 0) {
            vat.flux(ilk, address(this), usr, lot);
            _remove(id);
        } else {
            sales[id].tab = tab;
            sales[id].lot = lot;
        }

        emit Take(id, max, price, owe, tab, lot, usr);
    }
```

2.4.21 _remove(id)

```
    function _remove(uint256 id) internal {
        uint256 _move    = active[active.length - 1];
        if (id != _move) {
            uint256 _index    = sales[id].pos;
            active[_index]    = _move;
            sales[_move].pos = _index;
        }
        active.pop();
        delete sales[id];
    }
```

2.4.22 count()

```
    // The number of active auctions
    function count() external view returns (uint256) {
        return active.length;
    }
```

2.4.23 list()

```
    // Return the entire array of active auctions
    function list() external view returns (uint256[] memory) {
        return active;
    }
```

2.4.24 getStatus(id)

```
    // Externally returns boolean for if an auction needs a redo and also the
       ↪ current price
    function getStatus(uint256 id) external view returns (bool needsRedo,
       ↪ uint256 price, uint256 lot, uint256 tab) {
        // Read auction data
        address usr = sales[id].usr;
        uint96  tic = sales[id].tic;

        bool done;
        (done, price) = status(tic, sales[id].top);

        needsRedo = usr != address(0) && done;
        lot = sales[id].lot;
        tab = sales[id].tab;
    }
```

2.4.25 status(tic, top)

```
    // Internally returns boolean for if an auction needs a redo
    function status(uint96 tic, uint256 top) internal view returns (bool done,
       ↪ uint256 price) {
        price = calc.price(top, sub(block.timestamp, tic));
        done  = (sub(block.timestamp, tic) > tail || rdiv(price, top) < cusp);
    }
```

2.4.26 upchost() X

```
    // Public function to update the cached dust*chop value.
    function upchost() external {
        (,,,, uint256 _dust) = VatLike(vat).ilks(ilk);
        chost = wmul(_dust, dog.chop(ilk));
    }
```

2.4.27 yank(id) X a

```
    // Cancel an auction during ES or via governance action.
    function yank(uint256 id) external auth lock {
        require(sales[id].usr != address(0), "Clipper/not-running-auction");
        dog.digs(ilk, sales[id].tab);
        vat.flux(ilk, address(this), msg.sender, sales[id].lot);
        _remove(id);
        emit Yank(id);
    }
```

2.5 contract Dog

```
contract Dog {
    // --- Auth ---
    mapping (address => uint256) public wards;

    VatLike immutable public vat;  // CDP Engine

    mapping (bytes32 => Ilk) public ilks;

    VowLike public vow;   // Debt Engine
    uint256 public live;  // Active Flag
    uint256 public Hole;  // Max DAI needed to cover debt+fees of active
        ↪ auctions [rad]
    uint256 public Dirt;  // Amt DAI needed to cover debt+fees of active
        ↪ auctions [rad]

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);

    event File(bytes32 indexed what, uint256 data);
    event File(bytes32 indexed what, address data);
    event File(bytes32 indexed ilk, bytes32 indexed what, uint256 data);
    event File(bytes32 indexed ilk, bytes32 indexed what, address clip);

    event Bark(
      bytes32 indexed ilk,
      address indexed urn,
      uint256 ink,
      uint256 art,
      uint256 due,
      address clip,
      uint256 indexed id
    );
    event Digs(bytes32 indexed ilk, uint256 rad);
    event Cage();

    // --- Math ---
    uint256 constant WAD = 10 ** 18;
}
```

2.5.1 struct Dog.Ilk

```
    // --- Data ---
    struct Ilk {
        address clip;  // Liquidator
        uint256 chop;  // Liquidation Penalty
            ↪                                          [wad]
        uint256 hole;  // Max DAI needed to cover debt+fees of active auctions
            ↪ per ilk [rad]
        uint256 dirt;  // Amt DAI needed to cover debt+fees of active auctions
            ↪ per ilk [rad]
    }
```

2.5.2 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "Dog/not-authorized");
        _;
    }
```

2.5.3 rely(usr) X a

```
    function rely(address usr) external auth { wards[usr] = 1; emit Rely(usr); }
```

2.5.4 deny(usr) X a

```
    function deny(address usr) external auth { wards[usr] = 0; emit Deny(usr); }
```

2.5.5 constructor(vat_) X

```
    // --- Init ---
    constructor(address vat_) public {
        vat = VatLike(vat_);
        live = 1;
        wards[msg.sender] = 1;
        emit Rely(msg.sender);
    }
```

2.5.6 min(x, y)

```
    function min(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = x <= y ? x : y;
    }
```

2.5.7 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x);
    }
```

2.5.8 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x);
    }
```

2.5.9 mul(x, y)

```
    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require(y == 0 || (z = x * y) / y == x);
    }
```

2.5.10 file(what, data) X a

```
    // --- Administration ---
    function file(bytes32 what, address data) external auth {
        if (what == "vow") vow = VowLike(data);
        else revert("Dog/file-unrecognized-param");
        emit File(what, data);
    }
```

2.5.11 file(what, data) X a

```
    function file(bytes32 what, uint256 data) external auth {
        if (what == "Hole") Hole = data;
        else revert("Dog/file-unrecognized-param");
        emit File(what, data);
    }
```

2.5.12 file(ilk, what, data) X a

```
function file(bytes32 ilk, bytes32 what, uint256 data) external auth {
    if (what == "chop") {
        require(data >= WAD, "Dog/file-chop-lt-WAD");
        ilks[ilk].chop = data;
    } else if (what == "hole") ilks[ilk].hole = data;
    else revert("Dog/file-unrecognized-param");
    emit File(ilk, what, data);
}
```

2.5.13 file(ilk, what, clip) X a

```
function file(bytes32 ilk, bytes32 what, address clip) external auth {
    if (what == "clip") {
        require(ilk == ClipperLike(clip).ilk(), "Dog/file-ilk-neq-clip.ilk")
            ↪ ;
        ilks[ilk].clip = clip;
    } else revert("Dog/file-unrecognized-param");
    emit File(ilk, what, clip);
}
```

2.5.14 chop(ilk)

```
function chop(bytes32 ilk) external view returns (uint256) {
    return ilks[ilk].chop;
}
```

2.5.15 bark(ilk, urn, kpr) X

```
// --- CDP Liquidation: all bark and no bite ---
//
// Liquidate a Vault and start a Dutch auction to sell its collateral for
   ↪ DAI.
//
// The third argument is the address that will receive the liquidation
   ↪ reward, if any.
//
// The entire Vault will be liquidated except when the target amount of DAI
   ↪ to be raised in
// the resulting auction (debt of Vault + liquidation penalty) causes either
   ↪  Dirt to exceed
// Hole or ilk.dirt to exceed ilk.hole by an economically significant amount
   ↪ . In that
// case, a partial liquidation is performed to respect the global and per-
   ↪ ilk limits on
// outstanding DAI target. The one exception is if the resulting auction
   ↪ would likely
// have too little collateral to be interesting to Keepers (debt taken from
   ↪ Vault < ilk.dust),
// in which case the function reverts. Please refer to the code and comments
   ↪  within if
// more detail is desired.
function bark(bytes32 ilk, address urn, address kpr) external returns (
   ↪ uint256 id) {
    require(live == 1, "Dog/not-live");

    (uint256 ink, uint256 art) = vat.urns(ilk, urn);
    Ilk memory milk = ilks[ilk];
    uint256 dart;
    uint256 rate;
    uint256 dust;
    {
        uint256 spot;
```

```
            (,rate, spot,, dust) = vat.ilks(ilk);
            require(spot > 0 && mul(ink, spot) < mul(art, rate), "Dog/not-unsafe
                ↪ ");

            // Get the minimum value between:
            // 1) Remaining space in the general Hole
            // 2) Remaining space in the collateral hole
            require(Hole > Dirt && milk.hole > milk.dirt, "Dog/liquidation-limit
                ↪ -hit");
            uint256 room = min(Hole - Dirt, milk.hole - milk.dirt);

            // uint256.max()/(RAD*WAD) = 115,792,089,237,316
            dart = min(art, mul(room, WAD) / rate / milk.chop);

            // Partial liquidation edge case logic
            if (art > dart) {
                if (mul(art - dart, rate) < dust) {

                    // If the leftover Vault would be dusty, just liquidate it
                        ↪ entirely.
                    // This will result in at least one of dirt_i > hole_i or
                        ↪ Dirt > Hole becoming true.
                    // The amount of excess will be bounded above by ceiling(
                        ↪ dust_i * chop_i / WAD).
                    // This deviation is assumed to be small compared to both
                        ↪ hole_i and Hole, so that
                    // the extra amount of target DAI over the limits intended
                        ↪ is not of economic concern.
                    dart = art;
                } else {

                    // In a partial liquidation, the resulting auction should
                        ↪ also be non-dusty.
                    require(mul(dart, rate) >= dust, "Dog/dusty-auction-from-
                        ↪ partial-liquidation");
                }
            }
        }

        uint256 dink = mul(ink, dart) / art;

        require(dink > 0, "Dog/null-auction");
        require(dart <= 2**255 && dink <= 2**255, "Dog/overflow");

        vat.grab(
            ilk, urn, milk.clip, address(vow), -int256(dink), -int256(dart)
        );

        uint256 due = mul(dart, rate);
        vow.fess(due);

        {   // Avoid stack too deep
            // This calcuation will overflow if dart*rate exceeds ~10^14
            uint256 tab = mul(due, milk.chop) / WAD;
            Dirt = add(Dirt, tab);
            ilks[ilk].dirt = add(milk.dirt, tab);

            id = ClipperLike(milk.clip).kick({
                tab: tab,
                lot: dink,
                usr: urn,
                kpr: kpr
            });
        }

        emit Bark(ilk, urn, dink, dart, due, milk.clip, id);
    }
```

2.5.16 digs(ilk, rad) X a

```
    function digs(bytes32 ilk, uint256 rad) external auth {
        Dirt = sub(Dirt, rad);
        ilks[ilk].dirt = sub(ilks[ilk].dirt, rad);
        emit Digs(ilk, rad);
    }
```

2.5.17 cage() X a

```
    function cage() external auth {
        live = 0;
        emit Cage();
    }
```

2.6 contract ClipperMom

```
contract ClipperMom {
    address public owner;
    address public authority;
    mapping (address => uint256) public locked;    // timestamp when becomes
        ↪ unlocked (per clipper)
    mapping (address => uint256) public tolerance; // clipper -> ray

    SpotterLike public immutable spotter;

    event SetOwner(address indexed oldOwner, address indexed newOwner);
    event SetAuthority(address indexed oldAuthority, address indexed
        ↪ newAuthority);
    event SetBreaker(address indexed clip, uint256 level);

    // --- Math ---
    uint256 constant WAD = 10 ** 18;
    uint256 constant RAY = 10 ** 27;
}
```

2.6.1 modifier onlyOwner()

```
    modifier onlyOwner {
        require(msg.sender == owner, "ClipperMom/only-owner");
        _;
    }
```

2.6.2 modifier auth()

```
    modifier auth {
        require(isAuthorized(msg.sender, msg.sig), "ClipperMom/not-authorized");
        _;
    }
```

2.6.3 constructor(spotter_) X

```
    constructor(address spotter_) public {
        owner = msg.sender;
        spotter = SpotterLike(spotter_);
        emit SetOwner(address(0), msg.sender);
    }
```

2.6.4 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x);
    }
```

2.6.5 mul(x, y)

```
    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require(y == 0 || (z = x * y) / y == x);
    }
```

2.6.6 rmul(x, y)

```
    function rmul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = mul(x, y) / RAY;
    }
```

2.6.7 isAuthorized(src, sig)

```
function isAuthorized(address src, bytes4 sig) internal view returns (bool)
    ↪ {
    if (src == address(this)) {
        return true;
    } else if (src == owner) {
        return true;
    } else if (authority == address(0)) {
        return false;
    } else {
        return AuthorityLike(authority).canCall(src, address(this), sig);
    }
}
```

2.6.8 getPrices(clip)

```
function getPrices(address clip) internal view returns (uint256 cur, uint256
    ↪  nxt) {
    (OsmLike osm, ) = spotter.ilks(ClipLike(clip).ilk());
    bool has;
    (cur, has) = osm.peek();
    require(has, "ClipperMom/invalid-cur-price");
    (nxt, has) = osm.peep();
    require(has, "ClipperMom/invalid-nxt-price");
}
```

2.6.9 setOwner(owner_) X

```
// Governance actions with delay
function setOwner(address owner_) external onlyOwner {
    emit SetOwner(owner, owner_);
    owner = owner_;
}
```

2.6.10 setAuthority(authority_) X

```
function setAuthority(address authority_) external onlyOwner {
    emit SetAuthority(authority, authority_);
    authority = authority_;
}
```

2.6.11 setPriceTolerance(clip, value) X

```
// Set the price tolerance for a specific ilk.
// The price tolerance is the minimum acceptable value a new price can have
    ↪ relative to the previous price
// For instance, a tolerance of 0.6 means that a new price can't be lower
    ↪ than 60% of the previous price
// 0.6 * RAY = 600000000000000000000000000 => means acceptable drop from
    ↪ previous price is up to 40%
function setPriceTolerance(address clip, uint256 value) external onlyOwner {
    require(value <= 1 * RAY, "ClipperMom/tolerance-out-of-bounds");
    tolerance[clip] = value;
}
```

2.6.12 setBreaker(clip, level, delay) X a

```
    // Governance action without delay
    function setBreaker(address clip, uint256 level, uint256 delay) external
        ↪ auth {
        require(level <= 3, "ClipperMom/nonexistent-level");
        ClipLike(clip).file("stopped", level);
        // If governance changes the status of the breaker we want to lock for
            ↪ one hour
        // the permissionless function so the osm can pull new nxt price to
            ↪ compare
        locked[clip] = add(block.timestamp, delay);
        emit SetBreaker(clip, level);
    }
```

2.6.13 tripBreaker(clip) X

```
    /**
        The following implements a permissionless circuit breaker in case the
            ↪ price reported by an oracle
        for a particular collateral type will drop below than a governance-
            ↪ defined % from 1 hour to the next.

        The setPriceTolerance function sets that % (as a value between 0 and RAY
            ↪ ) for a specific collateral type.

        tripBreaker takes the address of some ilk's Clipper.
        It then gets the current and next price and checks whether the next
            ↪ price is less than the minimum
        acceptable next price based on the tolerance. If the next price is
            ↪ unacceptable (lower than rmul(current_price, tolerance)),
        it stops creation of new auctions and resets of current auctions for the
            ↪  Clipper's ilk. Currently, governance
        must reset the breaker manually.
    */
    function tripBreaker(address clip) external {
        require(ClipLike(clip).stopped() < 2, "ClipperMom/clipper-already-
            ↪ stopped");
        require(block.timestamp > locked[clip], "ClipperMom/temporary-locked");

        (uint256 cur, uint256 nxt) = getPrices(clip);

        // tolerance[clip] == 0 will always make the following require to revert
        require(nxt < rmul(cur, tolerance[clip]), "ClipperMom/price-within-
            ↪ bounds");
        ClipLike(clip).file("stopped", 2);
        emit SetBreaker(clip, 2);
    }
```

Chapter 3

System Stabilizer

3.1 contract Flapper

```
/*
   This thing lets you sell some dai in return for gems.

 - `lot` dai in return for bid
 - `bid` gems paid
 - `ttl` single bid lifetime
 - `beg` minimum bid increase
 - `end` max auction duration
*/

contract Flapper {
    // --- Auth ---
    mapping (address => uint) public wards;

    mapping (uint => Bid) public bids;

    VatLike  public   vat;  // CDP Engine
    GemLike  public   gem;

    uint256  constant ONE = 1.00E18;
    uint256  public   beg = 1.05E18;  // 5% minimum bid increase
    uint48   public   ttl = 3 hours;  // 3 hours bid duration         [seconds]
    uint48   public   tau = 2 days;   // 2 days total auction length  [seconds]
    uint256  public kicks = 0;
    uint256  public live;  // Active Flag
    uint256  public lid;   // max dai to be in auction at one time  [rad]
    uint256  public fill;  // current dai in auction                [rad]

    // --- Events ---
    event Kick(
      uint256 id,
      uint256 lot,
      uint256 bid
    );
}
```

3.1.1 struct Flapper.Bid

```
    // --- Data ---
    struct Bid {
        uint256 bid;  // gems paid                [wad]
        uint256 lot;  // dai in return for bid    [rad]
        address guy;  // high bidder
        uint48  tic;  // bid expiry time          [unix epoch time]
        uint48  end;  // auction expiry time      [unix epoch time]
    }
```

3.1.2 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "Flapper/not-authorized");
        _;
    }
```

3.1.3 rely(usr) X a

```
    function rely(address usr) external auth { wards[usr] = 1; }
```

3.1.4 deny(usr) X a

```
    function deny(address usr) external auth { wards[usr] = 0; }
```

3.1.5 constructor(vat_, gem_) X

```
    // --- Init ---
    constructor(address vat_, address gem_) public {
        wards[msg.sender] = 1;
        vat = VatLike(vat_);
        gem = GemLike(gem_);
        live = 1;
    }
```

3.1.6 add(x, y)

```
    // --- Math ---
    function add(uint48 x, uint48 y) internal pure returns (uint48 z) {
        require((z = x + y) >= x);
    }
```

3.1.7 add256(x, y)

```
    function add256(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x);
    }
```

3.1.8 sub(x, y)

```
    function sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x);
    }
```

3.1.9 mul(x, y)

```
    function mul(uint x, uint y) internal pure returns (uint z) {
        require(y == 0 || (z = x * y) / y == x);
    }
```

3.1.10 file(what, data) X a

```
    // --- Admin ---
    function file(bytes32 what, uint data) external auth {
        if (what == "beg") beg = data;
        else if (what == "ttl") ttl = uint48(data);
        else if (what == "tau") tau = uint48(data);
        else if (what == "lid") lid = data;
        else revert("Flapper/file-unrecognized-param");
    }
```

3.1.11 kick(lot, bid) X a

```
    // --- Auction ---
    function kick(uint lot, uint bid) external auth returns (uint id) {
        require(live == 1, "Flapper/not-live");
        require(kicks < uint(-1), "Flapper/overflow");
        fill = add256(fill, lot);
        require(fill <= lid, "Flapper/over-lid");
        id = ++kicks;

        bids[id].bid = bid;
        bids[id].lot = lot;
        bids[id].guy = msg.sender;  // configurable??
        bids[id].end = add(uint48(now), tau);

        vat.move(msg.sender, address(this), lot);

        emit Kick(id, lot, bid);
    }
```

3.1.12 tick(id) X

```
    function tick(uint id) external {
        require(bids[id].end < now, "Flapper/not-finished");
        require(bids[id].tic == 0, "Flapper/bid-already-placed");
        bids[id].end = add(uint48(now), tau);
    }
```

3.1.13 tend(id, lot, bid) X

```
    function tend(uint id, uint lot, uint bid) external {
        require(live == 1, "Flapper/not-live");
        require(bids[id].guy != address(0), "Flapper/guy-not-set");
        require(bids[id].tic > now || bids[id].tic == 0, "Flapper/already-
           ↪ finished-tic");
        require(bids[id].end > now, "Flapper/already-finished-end");

        require(lot == bids[id].lot, "Flapper/lot-not-matching");
        require(bid >  bids[id].bid, "Flapper/bid-not-higher");
        require(mul(bid, ONE) >= mul(beg, bids[id].bid), "Flapper/insufficient-
           ↪ increase");

        if (msg.sender != bids[id].guy) {
            gem.move(msg.sender, bids[id].guy, bids[id].bid);
            bids[id].guy = msg.sender;
        }
        gem.move(msg.sender, address(this), bid - bids[id].bid);

        bids[id].bid = bid;
        bids[id].tic = add(uint48(now), ttl);
    }
```

3.1.14 deal(id) X

```
    function deal(uint id) external {
        require(live == 1, "Flapper/not-live");
        require(bids[id].tic != 0 && (bids[id].tic < now || bids[id].end < now),
            ↪ "Flapper/not-finished");
        uint256 lot = bids[id].lot;
        vat.move(address(this), bids[id].guy, lot);
        gem.burn(address(this), bids[id].bid);
        delete bids[id];
        fill = sub(fill, lot);
    }
```

3.1.15 cage(rad) X a

```
    function cage(uint rad) external auth {
       live = 0;
       vat.move(address(this), msg.sender, rad);
    }
```

3.1.16 yank(id) X

```
    function yank(uint id) external {
        require(live == 0, "Flapper/still-live");
        require(bids[id].guy != address(0), "Flapper/guy-not-set");
        gem.move(address(this), bids[id].guy, bids[id].bid);
        delete bids[id];
    }
```

3.2 contract Flopper

```
/*
   This thing creates gems on demand in return for dai.

 - `lot` gems in return for bid
 - `bid` dai paid
 - `gal` receives dai income
 - `ttl` single bid lifetime
 - `beg` minimum bid increase
 - `end` max auction duration
*/

contract Flopper {
    // --- Auth ---
    mapping (address => uint) public wards;

    mapping (uint => Bid) public bids;

    VatLike  public   vat;  // CDP Engine
    GemLike  public   gem;

    uint256  constant ONE = 1.00E18;
    uint256  public   beg = 1.05E18;  // 5% minimum bid increase
    uint256  public   pad = 1.50E18;  // 50% lot increase for tick
    uint48   public   ttl = 3 hours;  // 3 hours bid lifetime         [seconds]
    uint48   public   tau = 2 days;   // 2 days total auction length  [seconds]
    uint256  public kicks = 0;
    uint256  public live;             // Active Flag
    address  public vow;              // not used until shutdown

    // --- Events ---
    event Kick(
      uint256 id,
      uint256 lot,
      uint256 bid,
      address indexed gal
    );
}
```

3.2.1 struct Flopper.Bid

```
    // --- Data ---
    struct Bid {
        uint256 bid;  // dai paid                [rad]
        uint256 lot;  // gems in return for bid  [wad]
        address guy;  // high bidder
        uint48  tic;  // bid expiry time         [unix epoch time]
        uint48  end;  // auction expiry time     [unix epoch time]
    }
```

3.2.2 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "Flopper/not-authorized");
        _;
    }
```

3.2.3 rely(usr) X a

```
    function rely(address usr) external auth { wards[usr] = 1; }
```

3.2.4 deny(usr) X a

```
    function deny(address usr) external auth { wards[usr] = 0; }
```

3.2.5 constructor(vat_, gem_) X

```
    // --- Init ---
    constructor(address vat_, address gem_) public {
        wards[msg.sender] = 1;
        vat = VatLike(vat_);
        gem = GemLike(gem_);
        live = 1;
    }
```

3.2.6 add(x, y)

```
    // --- Math ---
    function add(uint48 x, uint48 y) internal pure returns (uint48 z) {
        require((z = x + y) >= x);
    }
```

3.2.7 mul(x, y)

```
    function mul(uint x, uint y) internal pure returns (uint z) {
        require(y == 0 || (z = x * y) / y == x);
    }
```

3.2.8 min(x, y)

```
    function min(uint x, uint y) internal pure returns (uint z) {
        if (x > y) { z = y; } else { z = x; }
    }
```

3.2.9 file(what, data) X a

```
    // --- Admin ---
    function file(bytes32 what, uint data) external auth {
        if (what == "beg") beg = data;
        else if (what == "pad") pad = data;
        else if (what == "ttl") ttl = uint48(data);
        else if (what == "tau") tau = uint48(data);
        else revert("Flopper/file-unrecognized-param");
    }
```

3.2.10 kick(gal, lot, bid) X a

```
    // --- Auction ---
    function kick(address gal, uint lot, uint bid) external auth returns (uint
       ↪ id) {
        require(live == 1, "Flopper/not-live");
        require(kicks < uint(-1), "Flopper/overflow");
        id = ++kicks;

        bids[id].bid = bid;
        bids[id].lot = lot;
        bids[id].guy = gal;
        bids[id].end = add(uint48(now), tau);

        emit Kick(id, lot, bid, gal);
    }
```

3.2.11 tick(id) X

```
    function tick(uint id) external {
        require(bids[id].end < now, "Flopper/not-finished");
        require(bids[id].tic == 0, "Flopper/bid-already-placed");
        bids[id].lot = mul(pad, bids[id].lot) / ONE;
        bids[id].end = add(uint48(now), tau);
    }
```

3.2.12 dent(id, lot, bid) X

```
    function dent(uint id, uint lot, uint bid) external {
        require(live == 1, "Flopper/not-live");
        require(bids[id].guy != address(0), "Flopper/guy-not-set");
        require(bids[id].tic > now || bids[id].tic == 0, "Flopper/already-
            ↪ finished-tic");
        require(bids[id].end > now, "Flopper/already-finished-end");

        require(bid == bids[id].bid, "Flopper/not-matching-bid");
        require(lot <  bids[id].lot, "Flopper/lot-not-lower");
        require(mul(beg, lot) <= mul(bids[id].lot, ONE), "Flopper/insufficient-
            ↪ decrease");

        if (msg.sender != bids[id].guy) {
            vat.move(msg.sender, bids[id].guy, bid);

            // on first dent, clear as much Ash as possible
            if (bids[id].tic == 0) {
                uint Ash = VowLike(bids[id].guy).Ash();
                VowLike(bids[id].guy).kiss(min(bid, Ash));
            }

            bids[id].guy = msg.sender;
        }

        bids[id].lot = lot;
        bids[id].tic = add(uint48(now), ttl);
    }
```

3.2.13 deal(id) X

```
    function deal(uint id) external {
        require(live == 1, "Flopper/not-live");
        require(bids[id].tic != 0 && (bids[id].tic < now || bids[id].end < now),
            ↪  "Flopper/not-finished");
        gem.mint(bids[id].guy, bids[id].lot);
        delete bids[id];
    }
```

3.2.14 cage() X a

```
    // --- Shutdown ---
    function cage() external auth {
       live = 0;
       vow = msg.sender;
    }
```

3.2.15 yank(id) X

```
    function yank(uint id) external {
        require(live == 0, "Flopper/still-live");
        require(bids[id].guy != address(0), "Flopper/guy-not-set");
        vat.suck(vow, bids[id].guy, bids[id].bid);
        delete bids[id];
    }
```

3.3 contract Vow

```
contract Vow {
    // --- Auth ---
    mapping (address => uint) public wards;

    // --- Data ---
    VatLike public vat;         // CDP Engine
    FlapLike public flapper;    // Surplus Auction House
    FlopLike public flopper;    // Debt Auction House

    mapping (uint256 => uint256) public sin;  // debt queue
    uint256 public Sin;   // Queued debt            [rad]
    uint256 public Ash;   // On-auction debt        [rad]

    uint256 public wait;  // Flop delay             [seconds]
    uint256 public dump;  // Flop initial lot size  [wad]
    uint256 public sump;  // Flop fixed bid size    [rad]

    uint256 public bump;  // Flap fixed lot size    [rad]
    uint256 public hump;  // Surplus buffer         [rad]

    uint256 public live;  // Active Flag
}
```

3.3.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "Vow/not-authorized");
        _;
    }
```

3.3.2 rely(usr) X a

```
    function rely(address usr) external auth { require(live == 1, "Vow/not-live"
        ↪ ); wards[usr] = 1; }
```

3.3.3 deny(usr) X a

```
    function deny(address usr) external auth { wards[usr] = 0; }
```

3.3.4 constructor(vat_, flapper_, flopper_) X

```
    // --- Init ---
    constructor(address vat_, address flapper_, address flopper_) public {
        wards[msg.sender] = 1;
        vat     = VatLike(vat_);
        flapper = FlapLike(flapper_);
        flopper = FlopLike(flopper_);
        vat.hope(flapper_);
        live = 1;
    }
```

3.3.5 add(x, y)

```
    // --- Math ---
    function add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x);
    }
```

3.3.6 sub(x, y)

```
function sub(uint x, uint y) internal pure returns (uint z) {
    require((z = x - y) <= x);
}
```

3.3.7 min(x, y)

```
function min(uint x, uint y) internal pure returns (uint z) {
    return x <= y ? x : y;
}
```

3.3.8 file(what, data) X a

```
// --- Administration ---
function file(bytes32 what, uint data) external auth {
    if (what == "wait") wait = data;
    else if (what == "bump") bump = data;
    else if (what == "sump") sump = data;
    else if (what == "dump") dump = data;
    else if (what == "hump") hump = data;
    else revert("Vow/file-unrecognized-param");
}
```

3.3.9 file(what, data) X a

```
function file(bytes32 what, address data) external auth {
    if (what == "flapper") {
        vat.nope(address(flapper));
        flapper = FlapLike(data);
        vat.hope(data);
    }
    else if (what == "flopper") flopper = FlopLike(data);
    else revert("Vow/file-unrecognized-param");
}
```

3.3.10 fess(tab) X a

```
// Push to debt-queue
function fess(uint tab) external auth {
    sin[now] = add(sin[now], tab);
    Sin = add(Sin, tab);
}
```

3.3.11 flog(era) X

```
// Pop from debt-queue
function flog(uint era) external {
    require(add(era, wait) <= now, "Vow/wait-not-finished");
    Sin = sub(Sin, sin[era]);
    sin[era] = 0;
}
```

3.3.12 heal(rad) X

```
    // Debt settlement
    function heal(uint rad) external {
        require(rad <= vat.dai(address(this)), "Vow/insufficient-surplus");
        require(rad <= sub(sub(vat.sin(address(this)), Sin), Ash), "Vow/
           ↪ insufficient-debt");
        vat.heal(rad);
    }
```

3.3.13 kiss(rad) X

```
    function kiss(uint rad) external {
        require(rad <= Ash, "Vow/not-enough-ash");
        require(rad <= vat.dai(address(this)), "Vow/insufficient-surplus");
        Ash = sub(Ash, rad);
        vat.heal(rad);
    }
```

3.3.14 flop() X

```
    // Debt auction
    function flop() external returns (uint id) {
        require(sump <= sub(sub(vat.sin(address(this)), Sin), Ash), "Vow/
           ↪ insufficient-debt");
        require(vat.dai(address(this)) == 0, "Vow/surplus-not-zero");
        Ash = add(Ash, sump);
        id = flopper.kick(address(this), dump, sump);
    }
```

3.3.15 flap() X

```
    // Surplus auction
    function flap() external returns (uint id) {
        require(vat.dai(address(this)) >= add(add(vat.sin(address(this)), bump),
           ↪  hump), "Vow/insufficient-surplus");
        require(sub(sub(vat.sin(address(this)), Sin), Ash) == 0, "Vow/debt-not-
           ↪ zero");
        id = flapper.kick(bump, 0);
    }
```

3.3.16 cage() X a

```
    function cage() external auth {
        require(live == 1, "Vow/not-live");
        live = 0;
        Sin = 0;
        Ash = 0;
        flapper.cage(vat.dai(address(flapper)));
        flopper.cage();
        vat.heal(min(vat.dai(address(this)), vat.sin(address(this))));
    }
```

Chapter 4

Rates

4.1 contract Jug

```
contract Jug {
    // --- Auth ---
    mapping (address => uint) public wards;

    mapping (bytes32 => Ilk) public ilks;
    VatLike                  public vat;   // CDP Engine
    address                  public vow;   // Debt Engine
    uint256                  public base;  // Global, per-second stability fee
        ↪ contribution [ray]
    uint256 constant ONE = 10 ** 27;
}
```

4.1.1 struct Jug.Ilk

```
    // --- Data ---
    struct Ilk {
        uint256 duty;  // Collateral-specific, per-second stability fee
            ↪ contribution [ray]
        uint256  rho;  // Time of last drip [unix epoch time]
    }
```

4.1.2 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "Jug/not-authorized");
        _;
    }
```

4.1.3 rely(usr) X a

```
    function rely(address usr) external auth { wards[usr] = 1; }
```

4.1.4 deny(usr) X a

```
    function deny(address usr) external auth { wards[usr] = 0; }
```

4.1.5 constructor(vat_) X

```
    // --- Init ---
    constructor(address vat_) public {
        wards[msg.sender] = 1;
        vat = VatLike(vat_);
    }
```

4.1.6 _rpow(x, n, b)

```
    // --- Math ---
    function _rpow(uint x, uint n, uint b) internal pure returns (uint z) {
      assembly {
        switch x case 0 {switch n case 0 {z := b} default {z := 0}}
        default {
          switch mod(n, 2) case 0 { z := b } default { z := x }
          let half := div(b, 2)  // for rounding.
          for { n := div(n, 2) } n { n := div(n,2) } {
            let xx := mul(x, x)
            if iszero(eq(div(xx, x), x)) { revert(0,0) }
            let xxRound := add(xx, half)
            if lt(xxRound, xx) { revert(0,0) }
            x := div(xxRound, b)
            if mod(n,2) {
              let zx := mul(z, x)
              if and(iszero(iszero(x)), iszero(eq(div(zx, x), z))) { revert(0,0)
                  ↪  }
              let zxRound := add(zx, half)
              if lt(zxRound, zx) { revert(0,0) }
              z := div(zxRound, b)
            }
          }
        }
      }
    }
```

4.1.7 _add(x, y)

```
    function _add(uint x, uint y) internal pure returns (uint z) {
        z = x + y;
        require(z >= x);
    }
```

4.1.8 _diff(x, y)

```
    function _diff(uint x, uint y) internal pure returns (int z) {
        z = int(x) - int(y);
        require(int(x) >= 0 && int(y) >= 0);
    }
```

4.1.9 _rmul(x, y)

```
    function _rmul(uint x, uint y) internal pure returns (uint z) {
        z = x * y;
        require(y == 0 || z / y == x);
        z = z / ONE;
    }
```

4.1.10 init(ilk) X a

```
    // --- Administration ---
    function init(bytes32 ilk) external auth {
        Ilk storage i = ilks[ilk];
        require(i.duty == 0, "Jug/ilk-already-init");
        i.duty = ONE;
        i.rho  = now;
    }
```

4.1.11 file(ilk, what, data) X a

```
function file(bytes32 ilk, bytes32 what, uint data) external auth {
    require(now == ilks[ilk].rho, "Jug/rho-not-updated");
    if (what == "duty") ilks[ilk].duty = data;
    else revert("Jug/file-unrecognized-param");
}
```

4.1.12 file(what, data) X a

```
function file(bytes32 what, uint data) external auth {
    if (what == "base") base = data;
    else revert("Jug/file-unrecognized-param");
}
```

4.1.13 file(what, data) X a

```
function file(bytes32 what, address data) external auth {
    if (what == "vow") vow = data;
    else revert("Jug/file-unrecognized-param");
}
```

4.1.14 drip(ilk) X

```
// --- Stability Fee Collection ---
function drip(bytes32 ilk) external returns (uint rate) {
    require(now >= ilks[ilk].rho, "Jug/invalid-now");
    (, uint prev) = vat.ilks(ilk);
    rate = _rmul(_rpow(_add(base, ilks[ilk].duty), now - ilks[ilk].rho, ONE)
       ↪ , prev);
    vat.fold(ilk, vow, _diff(rate, prev));
    ilks[ilk].rho = now;
}
```

4.2 contract Pot

```
contract Pot {
    // --- Auth ---
    mapping (address => uint) public wards;

    // --- Data ---
    mapping (address => uint256) public pie;  // Normalised Savings Dai [wad]

    uint256 public Pie;   // Total Normalised Savings Dai  [wad]
    uint256 public dsr;   // The Dai Savings Rate          [ray]
    uint256 public chi;   // The Rate Accumulator          [ray]

    VatLike public vat;   // CDP Engine
    address public vow;   // Debt Engine
    uint256 public rho;   // Time of last drip     [unix epoch time]

    uint256 public live;  // Active Flag

    // --- Math ---
    uint256 constant ONE = 10 ** 27;
}
```

4.2.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "Pot/not-authorized");
        _;
    }
```

4.2.2 rely(guy) X a

```
    function rely(address guy) external auth { wards[guy] = 1; }
```

4.2.3 deny(guy) X a

```
    function deny(address guy) external auth { wards[guy] = 0; }
```

4.2.4 constructor(vat_) X

```
    // --- Init ---
    constructor(address vat_) public {
        wards[msg.sender] = 1;
        vat = VatLike(vat_);
        dsr = ONE;
        chi = ONE;
        rho = now;
        live = 1;
    }
```

4.2.5 _rpow(x, n, base)

```
    function _rpow(uint x, uint n, uint base) internal pure returns (uint z) {
        assembly {
            switch x case 0 {switch n case 0 {z := base} default {z := 0}}
            default {
                switch mod(n, 2) case 0 { z := base } default { z := x }
                let half := div(base, 2)  // for rounding.
                for { n := div(n, 2) } n { n := div(n,2) } {
                    let xx := mul(x, x)
                    if iszero(eq(div(xx, x), x)) { revert(0,0) }
                    let xxRound := add(xx, half)
                    if lt(xxRound, xx) { revert(0,0) }
                    x := div(xxRound, base)
                    if mod(n,2) {
                        let zx := mul(z, x)
                        if and(iszero(iszero(x)), iszero(eq(div(zx, x), z))) {
                            ↪ revert(0,0) }
                        let zxRound := add(zx, half)
                        if lt(zxRound, zx) { revert(0,0) }
                        z := div(zxRound, base)
                    }
                }
            }
        }
    }
```

4.2.6 _rmul(x, y)

```
    function _rmul(uint x, uint y) internal pure returns (uint z) {
        z = _mul(x, y) / ONE;
    }
```

4.2.7 _add(x, y)

```
    function _add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x);
    }
```

4.2.8 _sub(x, y)

```
    function _sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x);
    }
```

4.2.9 _mul(x, y)

```
    function _mul(uint x, uint y) internal pure returns (uint z) {
        require(y == 0 || (z = x * y) / y == x);
    }
```

4.2.10 file(what, data) X a

```
    // --- Administration ---
    function file(bytes32 what, uint256 data) external auth {
        require(live == 1, "Pot/not-live");
        require(now == rho, "Pot/rho-not-updated");
        if (what == "dsr") dsr = data;
        else revert("Pot/file-unrecognized-param");
    }
```

4.2.11 file(what, addr) X a

```
    function file(bytes32 what, address addr) external auth {
        if (what == "vow") vow = addr;
        else revert("Pot/file-unrecognized-param");
    }
```

4.2.12 cage() X a

```
function cage() external auth {
    live = 0;
    dsr = ONE;
}
```

4.2.13 drip() X

```
// --- Savings Rate Accumulation ---
function drip() external returns (uint tmp) {
    require(now >= rho, "Pot/invalid-now");
    tmp = _rmul(_rpow(dsr, now - rho, ONE), chi);
    uint chi_ = _sub(tmp, chi);
    chi = tmp;
    rho = now;
    vat.suck(address(vow), address(this), _mul(Pie, chi_));
}
```

4.2.14 join(wad) X

```
// --- Savings Dai Management ---
function join(uint wad) external {
    require(now == rho, "Pot/rho-not-updated");
    pie[msg.sender] = _add(pie[msg.sender], wad);
    Pie             = _add(Pie,             wad);
    vat.move(msg.sender, address(this), _mul(chi, wad));
}
```

4.2.15 exit(wad) X

```
function exit(uint wad) external {
    pie[msg.sender] = _sub(pie[msg.sender], wad);
    Pie             = _sub(Pie,             wad);
    vat.move(address(this), msg.sender, _mul(chi, wad));
}
```

Chapter 5

OSM

5.1 contract Median

```
contract Median is LibNote {

    // --- Auth ---
    mapping (address => uint) public wards;

    uint128        val;
    uint32  public age;
    bytes32 public constant wat = "ethusd"; // You want to change this every
       ↪ deploy
    uint256 public bar = 1;

    // Authorized oracles, set by an auth
    mapping (address => uint256) public orcl;

    // Whitelisted contracts, set by an auth
    mapping (address => uint256) public bud;

    // Mapping for at most 256 oracles
    mapping (uint8 => address) public slot;

    event LogMedianPrice(uint256 val, uint256 age);
}
```

Inherited:

```
contract LibNote {
    event LogNote(
        bytes4   indexed  sig,
        address  indexed  usr,
        bytes32  indexed  arg1,
        bytes32  indexed  arg2,
        bytes             data
    ) anonymous;
}
```

5.1.1 modifier note() [LibNote]

```
    modifier note {
        _;
        assembly {
            // log an 'anonymous' event with a constant 6 words of calldata
            // and four indexed topics: selector, caller, arg1 and arg2
            let mark := msize()                       // end of memory ensures
                ↪  zero
            mstore(0x40, add(mark, 288))            // update free memory
                ↪ pointer
            mstore(mark, 0x20)                      // bytes type data offset
            mstore(add(mark, 0x20), 224)            // bytes size (padded)
            calldatacopy(add(mark, 0x40), 0, 224)   // bytes payload
            log4(mark, 288,                              // calldata
                 shl(224, shr(224, calldataload(0))), // msg.sig
                 caller(),                              // msg.sender
                 calldataload(4),                     // arg1
                 calldataload(36)                     // arg2
                )
        }
    }
```

5.1.2 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "Median/not-authorized");
        _;
    }
```

5.1.3 modifier toll()

```
    modifier toll { require(bud[msg.sender] == 1, "Median/contract-not-
        ↪ whitelisted"); _;}
```

5.1.4 rely(usr) X a

```
    function rely(address usr) external note auth { wards[usr] = 1; }
```

5.1.5 deny(usr) X a

```
    function deny(address usr) external note auth { wards[usr] = 0; }
```

5.1.6 constructor() X

```
    //Set type of Oracle
    constructor() public {
        wards[msg.sender] = 1;
    }
```

5.1.7 read()

```
    function read() external view toll returns (uint256) {
        require(val > 0, "Median/invalid-price-feed");
        return val;
    }
```

5.1.8 peek()

```
    function peek() external view toll returns (uint256,bool) {
        return (val, val > 0);
    }
```

5.1.9 recover(val_, age_, v, r, s)

```
function recover(uint256 val_, uint256 age_, uint8 v, bytes32 r, bytes32 s)
   ↪ internal pure returns (address) {
    return ecrecover(
        keccak256(abi.encodePacked("\x19Ethereum Signed Message:\n32",
            ↪ keccak256(abi.encodePacked(val_, age_, wat)))),
        v, r, s
    );
}
```

5.1.10 poke(val_, age_, v, r, s) X

```
function poke(
    uint256[] calldata val_, uint256[] calldata age_,
    uint8[] calldata v, bytes32[] calldata r, bytes32[] calldata s) external
{
    require(val_.length == bar, "Median/bar-too-low");

    uint256 bloom = 0;
    uint256 last = 0;
    uint256 zzz = age;

    for (uint i = 0; i < val_.length; i++) {
        // Validate the values were signed by an authorized oracle
        address signer = recover(val_[i], age_[i], v[i], r[i], s[i]);
        // Check that signer is an oracle
        require(orcl[signer] == 1, "Median/invalid-oracle");
        // Price feed age greater than last medianizer age
        require(age_[i] > zzz, "Median/stale-message");
        // Check for ordered values
        require(val_[i] >= last, "Median/messages-not-in-order");
        last = val_[i];
        // Bloom filter for signer uniqueness
        uint8 sl = uint8(uint256(signer) >> 152);
        require((bloom >> sl) % 2 == 0, "Median/oracle-already-signed");
        bloom += uint256(2) ** sl;
    }

    val = uint128(val_[val_.length >> 1]);
    age = uint32(block.timestamp);

    emit LogMedianPrice(val, age);
}
```

5.1.11 lift(a) X a

```
function lift(address[] calldata a) external note auth {
    for (uint i = 0; i < a.length; i++) {
        require(a[i] != address(0), "Median/no-oracle-0");
        uint8 s = uint8(uint256(a[i]) >> 152);
        require(slot[s] == address(0), "Median/signer-already-exists");
        orcl[a[i]] = 1;
        slot[s] = a[i];
    }
}
```

5.1.12 drop(a) X a

```
function drop(address[] calldata a) external note auth {
   for (uint i = 0; i < a.length; i++) {
        orcl[a[i]] = 0;
        slot[uint8(uint256(a[i]) >> 152)] = address(0);
   }
```

```
    }
```

5.1.13 setBar(bar_) X a

```
    function setBar(uint256 bar_) external note auth {
        require(bar_ > 0, "Median/quorum-is-zero");
        require(bar_ % 2 != 0, "Median/quorum-not-odd-number");
        bar = bar_;
    }
```

5.1.14 kiss(a) X a

```
    function kiss(address a) external note auth {
        require(a != address(0), "Median/no-contract-0");
        bud[a] = 1;
    }
```

5.1.15 diss(a) X a

```
    function diss(address a) external note auth {
        bud[a] = 0;
    }
```

5.1.16 kiss(a) X a

```
    function kiss(address[] calldata a) external note auth {
        for(uint i = 0; i < a.length; i++) {
            require(a[i] != address(0), "Median/no-contract-0");
            bud[a[i]] = 1;
        }
    }
```

5.1.17 diss(a) X a

```
    function diss(address[] calldata a) external note auth {
        for(uint i = 0; i < a.length; i++) {
            bud[a[i]] = 0;
        }
    }
```

5.2 contract OSM

```
contract OSM is LibNote {

    // --- Auth ---
    mapping (address => uint) public wards;

    // --- Stop ---
    uint256 public stopped;

    address public src;
    uint16  constant ONE_HOUR = uint16(3600);
    uint16  public hop = ONE_HOUR;
    uint64  public zzz;

    Feed cur;
    Feed nxt;

    // Whitelisted contracts, set by an auth
    mapping (address => uint256) public bud;

    event LogValue(bytes32 val);
}
```

Inherited:

```
contract LibNote {
    event LogNote(
        bytes4   indexed  sig,
        address  indexed  usr,
        bytes32  indexed  arg1,
        bytes32  indexed  arg2,
        bytes             data
    ) anonymous;
}
```

5.2.1 struct OSM.Feed

```
    struct Feed {
        uint128 val;
        uint128 has;
    }
```

5.2.2 modifier note() [LibNote(2)]

```
    modifier note {
        _;
        assembly {
            // log an 'anonymous' event with a constant 6 words of calldata
            // and four indexed topics: selector, caller, arg1 and arg2
            let mark := msize()                          // end of memory ensures
                ↪ zero
            mstore(0x40, add(mark, 288))                 // update free memory
                ↪ pointer
            mstore(mark, 0x20)                           // bytes type data offset
            mstore(add(mark, 0x20), 224)                 // bytes size (padded)
            calldatacopy(add(mark, 0x40), 0, 224)        // bytes payload
            log4(mark, 288,                              // calldata
                 shl(224, shr(224, calldataload(0))),    // msg.sig
                 caller(),                               // msg.sender
                 calldataload(4),                        // arg1
                 calldataload(36)                        // arg2
                )
        }
    }
```

5.2.3 modifier auth()

```
modifier auth {
    require(wards[msg.sender] == 1, "OSM/not-authorized");
    _;
}
```

5.2.4 modifier stoppable()

```
modifier stoppable { require(stopped == 0, "OSM/is-stopped"); _; }
```

5.2.5 modifier toll()

```
modifier toll { require(bud[msg.sender] == 1, "OSM/contract-not-whitelisted"
    ↪ ); _; }
```

5.2.6 rely(usr) X a

```
function rely(address usr) external note auth { wards[usr] = 1; }
```

5.2.7 deny(usr) X a

```
function deny(address usr) external note auth { wards[usr] = 0; }
```

5.2.8 add(x, y)

```
// --- Math ---
function add(uint64 x, uint64 y) internal pure returns (uint64 z) {
    z = x + y;
    require(z >= x);
}
```

5.2.9 constructor(src_) X

```
constructor (address src_) public {
    wards[msg.sender] = 1;
    src = src_;
}
```

5.2.10 stop() X a

```
function stop() external note auth {
    stopped = 1;
}
```

5.2.11 start() X a

```
function start() external note auth {
    stopped = 0;
}
```

5.2.12 change(src_) X a

```
function change(address src_) external note auth {
    src = src_;
}
```

5.2.13 era()

```
function era() internal view returns (uint) {
    return block.timestamp;
}
```

5.2.14 prev(ts)

```
function prev(uint ts) internal view returns (uint64) {
    require(hop != 0, "OSM/hop-is-zero");
    return uint64(ts - (ts % hop));
}
```

5.2.15 step(ts) X a

```
function step(uint16 ts) external auth {
    require(ts > 0, "OSM/ts-is-zero");
    hop = ts;
}
```

5.2.16 void() X a

```
function void() external note auth {
    cur = nxt = Feed(0, 0);
    stopped = 1;
}
```

5.2.17 pass()

```
function pass() public view returns (bool ok) {
    return era() >= add(zzz, hop);
}
```

5.2.18 poke() X

```
function poke() external note stoppable {
    require(pass(), "OSM/not-passed");
    (bytes32 wut, bool ok) = DSValue(src).peek();
    if (ok) {
        cur = nxt;
        nxt = Feed(uint128(uint(wut)), 1);
        zzz = prev(era());
        emit LogValue(bytes32(uint(cur.val)));
    }
}
```

5.2.19 peek()

```
function peek() external view toll returns (bytes32,bool) {
    return (bytes32(uint(cur.val)), cur.has == 1);
}
```

5.2.20 peep()

```
    function peep() external view toll returns (bytes32,bool) {
        return (bytes32(uint(nxt.val)), nxt.has == 1);
    }
```

5.2.21 read()

```
    function read() external view toll returns (bytes32) {
        require(cur.has == 1, "OSM/no-current-value");
        return (bytes32(uint(cur.val)));
    }
```

5.2.22 kiss(a) X a

```
    function kiss(address a) external note auth {
        require(a != address(0), "OSM/no-contract-0");
        bud[a] = 1;
    }
```

5.2.23 diss(a) X a

```
    function diss(address a) external note auth {
        bud[a] = 0;
    }
```

5.2.24 kiss(a) X a

```
    function kiss(address[] calldata a) external note auth {
        for(uint i = 0; i < a.length; i++) {
            require(a[i] != address(0), "OSM/no-contract-0");
            bud[a[i]] = 1;
        }
    }
```

5.2.25 diss(a) X a

```
    function diss(address[] calldata a) external note auth {
        for(uint i = 0; i < a.length; i++) {
            bud[a[i]] = 0;
        }
    }
```

5.3 contract Value

```
contract Value {
    bool     has;
    bytes32 val;
}
```

5.3.1 peek()

```
    function peek() public view returns (bytes32, bool) {
        return (val,has);
    }
```

Chapter 6

Emergency Shutdown

6.1 contract Cure

```
contract Cure {
    mapping (address => uint256) public wards;
    uint256 public live;
    address[] public srcs;
    uint256 public wait;
    uint256 public when;
    mapping (address => uint256) public pos; // position in srcs + 1, 0 means a
        ↪ source does not exist
    mapping (address => uint256) public amt;
    mapping (address => uint256) public loaded;
    uint256 public lCount;
    uint256 public say;

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event File(bytes32 indexed what, uint256 data);
    event Lift(address indexed src);
    event Drop(address indexed src);
    event Load(address indexed src);
    event Cage();
}
```

6.1.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "Cure/not-authorized");
        _;
    }
```

6.1.2 _add(x, y)

```
    // --- Internal ---
    function _add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "Cure/add-overflow");
    }
```

6.1.3 _sub(x, y)

```
    function _sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "Cure/sub-underflow");
    }
```

6.1.4 constructor() X

```
    constructor() public {
        live = 1;
        wards[msg.sender] = 1;
        emit Rely(msg.sender);
    }
```

6.1.5 tCount()

```
    function tCount() external view returns (uint256 count_) {
        count_ = srcs.length;
    }
```

6.1.6 list()

```
    function list() external view returns (address[] memory) {
        return srcs;
    }
```

6.1.7 tell()

```
    function tell() external view returns (uint256) {
        require(live == 0 && (lCount == srcs.length || block.timestamp >= when),
            ↪ "Cure/missing-load-and-time-not-passed");
        return say;
    }
```

6.1.8 rely(usr) X a

```
    function rely(address usr) external auth {
        require(live == 1, "Cure/not-live");
        wards[usr] = 1;
        emit Rely(usr);
    }
```

6.1.9 deny(usr) X a

```
    function deny(address usr) external auth {
        require(live == 1, "Cure/not-live");
        wards[usr] = 0;
        emit Deny(usr);
    }
```

6.1.10 file(what, data) X a

```
    function file(bytes32 what, uint256 data) external auth {
        require(live == 1, "Cure/not-live");
        if (what == "wait") wait = data;
        else revert("Cure/file-unrecognized-param");
        emit File(what, data);
    }
```

6.1.11 lift(src) X a

```
function lift(address src) external auth {
    require(live == 1, "Cure/not-live");
    require(pos[src] == 0, "Cure/already-existing-source");
    srcs.push(src);
    pos[src] = srcs.length;
    emit Lift(src);
}
```

6.1.12 drop(src) X a

```
function drop(address src) external auth {
    require(live == 1, "Cure/not-live");
    uint256 pos_ = pos[src];
    require(pos_ > 0, "Cure/non-existing-source");
    uint256 last = srcs.length;
    if (pos_ < last) {
        address move = srcs[last - 1];
        srcs[pos_ - 1] = move;
        pos[move] = pos_;
    }
    srcs.pop();
    delete pos[src];
    delete amt[src];
    emit Drop(src);
}
```

6.1.13 cage() X a

```
function cage() external auth {
    require(live == 1, "Cure/not-live");
    live = 0;
    when = _add(block.timestamp, wait);
    emit Cage();
}
```

6.1.14 load(src) X

```
function load(address src) external {
    require(live == 0, "Cure/still-live");
    require(pos[src] > 0, "Cure/non-existing-source");
    uint256 oldAmt_ = amt[src];
    uint256 newAmt_ = amt[src] = SourceLike(src).cure();
    say = _add(_sub(say, oldAmt_), newAmt_);
    if (loaded[src] == 0) {
        loaded[src] = 1;
        lCount++;
    }
    emit Load(src);
}
```

6.2 contract End

```
/*
    This is the `End` and it coordinates Global Settlement. This is an
    involved, stateful process that takes place over nine steps.

    First we freeze the system and lock the prices for each ilk.

    1. `cage()`:
        - freezes user entrypoints
        - cancels flop/flap auctions
        - starts cooldown period
        - stops pot drips

    2. `cage(ilk)`:
       - set the cage price for each `ilk`, reading off the price feed

    We must process some system state before it is possible to calculate
    the final dai / collateral price. In particular, we need to determine

      a. `gap`, the collateral shortfall per collateral type by
         considering under-collateralised CDPs.

      b. `debt`, the outstanding dai supply after including system
         surplus / deficit

    We determine (a) by processing all under-collateralised CDPs with
    `skim`:

    3. `skim(ilk, urn)`:
       - cancels CDP debt
       - any excess collateral remains
       - backing collateral taken

    We determine (b) by processing ongoing dai generating processes,
    i.e. auctions. We need to ensure that auctions will not generate any
    further dai income.

    In the two-way auction model (Flipper) this occurs when
    all auctions are in the reverse (`dent`) phase. There are two ways
    of ensuring this:

    4a. i) `wait`: set the cooldown period to be at least as long as the
           longest auction duration, which needs to be determined by the
           cage administrator.

           This takes a fairly predictable time to occur but with altered
           auction dynamics due to the now varying price of dai.

       ii) `skip`: cancel all ongoing auctions and seize the collateral.

           This allows for faster processing at the expense of more
           processing calls. This option allows dai holders to retrieve
           their collateral faster.

           `skip(ilk, id)`:
            - cancel individual flip auctions in the `tend` (forward) phase
            - retrieves collateral and debt (including penalty) to owner's CDP
            - returns dai to last bidder
            - `dent` (reverse) phase auctions can continue normally

    Option (i), `wait`, is sufficient (if all auctions were bidded at least
    once) for processing the system settlement but option (ii), `skip`,
    will speed it up. Both options are available in this implementation,
    with `skip` being enabled on a per-auction basis.

    In the case of the Dutch Auctions model (Clipper) they keep recovering
    debt during the whole lifetime and there isn't a max duration time
    guaranteed for the auction to end.
    So the way to ensure the protocol will not receive extra dai income is:

    4b. i) ‘snip‘: cancel all ongoing auctions and seize the collateral.

           ‘snip(ilk, id)‘:
            - cancel individual running clip auctions
            - retrieves remaining collateral and debt (including penalty)
              to owner's CDP

    When a CDP has been processed and has no debt remaining, the
    remaining collateral can be removed.

    5. ‘free(ilk)‘:
        - remove collateral from the caller's CDP
        - owner can call as needed

    After the processing period has elapsed, we enable calculation of
    the final price for each collateral type.

    6. ‘thaw()‘:
       - only callable after processing time period elapsed
       - assumption that all under-collateralised CDPs are processed
       - fixes the total outstanding supply of dai
       - may also require extra CDP processing to cover vow surplus

    7. ‘flow(ilk)‘:
        - calculate the ‘fix‘, the cash price for a given ilk
        - adjusts the ‘fix‘ in the case of deficit / surplus

    At this point we have computed the final price for each collateral
    type and dai holders can now turn their dai into collateral. Each
    unit dai can claim a fixed basket of collateral.

    Dai holders must first ‘pack‘ some dai into a ‘bag‘. Once packed,
    dai cannot be unpacked and is not transferrable. More dai can be
    added to a bag later.

    8. ‘pack(wad)‘:
        - put some dai into a bag in preparation for ‘cash‘

    Finally, collateral can be obtained with ‘cash‘. The bigger the bag,
    the more collateral can be released.

    9. ‘cash(ilk, wad)‘:
        - exchange some dai from your bag for gems from a specific ilk
        - the number of gems is limited by how big your bag is
*/

contract End {
    // --- Auth ---
    mapping (address => uint256) public wards;

    // --- Data ---
    VatLike  public vat;   // CDP Engine
    CatLike  public cat;
    DogLike  public dog;
    VowLike  public vow;   // Debt Engine
    PotLike  public pot;
    SpotLike public spot;
    CureLike public cure;

    uint256  public live;  // Active Flag
    uint256  public when;  // Time of cage                   [unix epoch time]
    uint256  public wait;  // Processing Cooldown Length             [seconds]
    uint256  public debt;  // Total outstanding dai following processing [rad]

    mapping (bytes32 => uint256) public tag;  // Cage price              [ray]
    mapping (bytes32 => uint256) public gap;  // Collateral shortfall    [wad]
    mapping (bytes32 => uint256) public Art;  // Total debt per ilk       [wad]
    mapping (bytes32 => uint256) public fix;  // Final cash price          [ray]

    mapping (address => uint256)                        public bag;  //    [wad]
    mapping (bytes32 => mapping (address => uint256)) public out;  //    [wad]

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);

    event File(bytes32 indexed what, uint256 data);
    event File(bytes32 indexed what, address data);

    event Cage();
    event Cage(bytes32 indexed ilk);
    event Snip(bytes32 indexed ilk, uint256 indexed id, address indexed usr,
        ↪ uint256 tab, uint256 lot, uint256 art);
    event Skip(bytes32 indexed ilk, uint256 indexed id, address indexed usr,
        ↪ uint256 tab, uint256 lot, uint256 art);
    event Skim(bytes32 indexed ilk, address indexed urn, uint256 wad, uint256
        ↪ art);
    event Free(bytes32 indexed ilk, address indexed usr, uint256 ink);
    event Thaw();
    event Flow(bytes32 indexed ilk);
    event Pack(address indexed usr, uint256 wad);
    event Cash(bytes32 indexed ilk, address indexed usr, uint256 wad);

    // --- Math ---
    uint256 constant WAD = 10 ** 18;
    uint256 constant RAY = 10 ** 27;
}
```

6.2.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "End/not-authorized");
        _;
    }
```

6.2.2 rely(usr) X a

```
    function rely(address usr) external auth { wards[usr] = 1; emit Rely(usr); }
```

6.2.3 deny(usr) X a

```
    function deny(address usr) external auth { wards[usr] = 0; emit Deny(usr); }
```

6.2.4 constructor() X

```
    // --- Init ---
    constructor() public {
        wards[msg.sender] = 1;
        live = 1;
        emit Rely(msg.sender);
    }
```

6.2.5 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = x + y;
        require(z >= x);
    }
```

6.2.6 sub(x, y)

```
function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
    require((z = x - y) <= x);
}
```

6.2.7 mul(x, y)

```
function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
    require(y == 0 || (z = x * y) / y == x);
}
```

6.2.8 min(x, y)

```
function min(uint256 x, uint256 y) internal pure returns (uint256 z) {
    return x <= y ? x : y;
}
```

6.2.9 rmul(x, y)

```
function rmul(uint256 x, uint256 y) internal pure returns (uint256 z) {
    z = mul(x, y) / RAY;
}
```

6.2.10 wdiv(x, y)

```
function wdiv(uint256 x, uint256 y) internal pure returns (uint256 z) {
    z = mul(x, WAD) / y;
}
```

6.2.11 file(what, data) X a

```
// --- Administration ---
function file(bytes32 what, address data) external auth {
    require(live == 1, "End/not-live");
    if (what == "vat")  vat = VatLike(data);
    else if (what == "cat")  cat = CatLike(data);
    else if (what == "dog")  dog = DogLike(data);
    else if (what == "vow")  vow = VowLike(data);
    else if (what == "pot")  pot = PotLike(data);
    else if (what == "spot") spot = SpotLike(data);
    else if (what == "cure") cure = CureLike(data);
    else revert("End/file-unrecognized-param");
    emit File(what, data);
}
```

6.2.12 file(what, data) X a

```
function file(bytes32 what, uint256 data) external auth {
    require(live == 1, "End/not-live");
    if (what == "wait") wait = data;
    else revert("End/file-unrecognized-param");
    emit File(what, data);
}
```

6.2.13 cage() X a

```
    // --- Settlement ---
    function cage() external auth {
        require(live == 1, "End/not-live");
        live = 0;
        when = block.timestamp;
        vat.cage();
        cat.cage();
        dog.cage();
        vow.cage();
        spot.cage();
        pot.cage();
        cure.cage();
        emit Cage();
    }
```

6.2.14 cage(ilk) X

```
    function cage(bytes32 ilk) external {
        require(live == 0, "End/still-live");
        require(tag[ilk] == 0, "End/tag-ilk-already-defined");
        (Art[ilk],,,,) = vat.ilks(ilk);
        (PipLike pip,) = spot.ilks(ilk);
        // par is a ray, pip returns a wad
        tag[ilk] = wdiv(spot.par(), uint256(pip.read()));
        emit Cage(ilk);
    }
```

6.2.15 snip(ilk, id) X

```
    function snip(bytes32 ilk, uint256 id) external {
        require(tag[ilk] != 0, "End/tag-ilk-not-defined");

        (address _clip,,,) = dog.ilks(ilk);
        ClipLike clip = ClipLike(_clip);
        (, uint256 rate,,,) = vat.ilks(ilk);
        (, uint256 tab, uint256 lot, address usr,,) = clip.sales(id);

        vat.suck(address(vow), address(vow),  tab);
        clip.yank(id);

        uint256 art = tab / rate;
        Art[ilk] = add(Art[ilk], art);
        require(int256(lot) >= 0 && int256(art) >= 0, "End/overflow");
        vat.grab(ilk, usr, address(this), address(vow), int256(lot), int256(art)
            ↪ );
        emit Snip(ilk, id, usr, tab, lot, art);
    }
```

6.2.16 skip(ilk, id) X

```
    function skip(bytes32 ilk, uint256 id) external {
        require(tag[ilk] != 0, "End/tag-ilk-not-defined");

        (address _flip,,) = cat.ilks(ilk);
        FlipLike flip = FlipLike(_flip);
        (, uint256 rate,,,) = vat.ilks(ilk);
        (uint256 bid, uint256 lot,,,, address usr,, uint256 tab) = flip.bids(id)
            ↪ ;

        vat.suck(address(vow), address(vow),  tab);
        vat.suck(address(vow), address(this), bid);
        vat.hope(address(flip));
        flip.yank(id);

        uint256 art = tab / rate;
        Art[ilk] = add(Art[ilk], art);
        require(int256(lot) >= 0 && int256(art) >= 0, "End/overflow");
        vat.grab(ilk, usr, address(this), address(vow), int256(lot), int256(art)
            ↪ );
        emit Skip(ilk, id, usr, tab, lot, art);
    }
```

6.2.17 skim(ilk, urn) X

```
    function skim(bytes32 ilk, address urn) external {
        require(tag[ilk] != 0, "End/tag-ilk-not-defined");
        (, uint256 rate,,,) = vat.ilks(ilk);
        (uint256 ink, uint256 art) = vat.urns(ilk, urn);

        uint256 owe = rmul(rmul(art, rate), tag[ilk]);
        uint256 wad = min(ink, owe);
        gap[ilk] = add(gap[ilk], sub(owe, wad));

        require(wad <= 2**255 && art <= 2**255, "End/overflow");
        vat.grab(ilk, urn, address(this), address(vow), -int256(wad), -int256(
            ↪ art));
        emit Skim(ilk, urn, wad, art);
    }
```

6.2.18 free(ilk) X

```
    function free(bytes32 ilk) external {
        require(live == 0, "End/still-live");
        (uint256 ink, uint256 art) = vat.urns(ilk, msg.sender);
        require(art == 0, "End/art-not-zero");
        require(ink <= 2**255, "End/overflow");
        vat.grab(ilk, msg.sender, msg.sender, address(vow), -int256(ink), 0);
        emit Free(ilk, msg.sender, ink);
    }
```

6.2.19 thaw() X

```
    function thaw() external {
        require(live == 0, "End/still-live");
        require(debt == 0, "End/debt-not-zero");
        require(vat.dai(address(vow)) == 0, "End/surplus-not-zero");
        require(block.timestamp >= add(when, wait), "End/wait-not-finished");
        debt = sub(vat.debt(), cure.tell());
        emit Thaw();
    }
```

6.2.20 flow(ilk) X

```
    function flow(bytes32 ilk) external {
        require(debt != 0, "End/debt-zero");
        require(fix[ilk] == 0, "End/fix-ilk-already-defined");

        (, uint256 rate,,,) = vat.ilks(ilk);
        uint256 wad = rmul(rmul(Art[ilk], rate), tag[ilk]);
        fix[ilk] = mul(sub(wad, gap[ilk]), RAY) / (debt / RAY);
        emit Flow(ilk);
    }
```

6.2.21 pack(wad) X

```
    function pack(uint256 wad) external {
        require(debt != 0, "End/debt-zero");
        vat.move(msg.sender, address(vow), mul(wad, RAY));
        bag[msg.sender] = add(bag[msg.sender], wad);
        emit Pack(msg.sender, wad);
    }
```

6.2.22 cash(ilk, wad) X

```
    function cash(bytes32 ilk, uint256 wad) external {
        require(fix[ilk] != 0, "End/fix-ilk-not-defined");
        vat.flux(ilk, address(this), msg.sender, rmul(wad, fix[ilk]));
        out[ilk][msg.sender] = add(out[ilk][msg.sender], wad);
        require(out[ilk][msg.sender] <= bag[msg.sender], "End/insufficient-bag-
            ↪ balance");
        emit Cash(ilk, msg.sender, wad);
    }
```

6.3 contract ESM

```
contract ESM {

    uint256 constant WAD = 10 ** 18;

    GemLike public immutable gem;   // collateral (MKR token)
    address public immutable proxy; // Pause proxy

    mapping(address => uint256) public wards; // auth
    mapping(address => uint256) public sum;   // per-address balance

    uint256 public Sum;  // total balance
    uint256 public min;  // minimum activation threshold [wad]
    EndLike public end;  // cage module
    uint256 public live; // active flag

    event Fire();
    event Join(address indexed usr, uint256 wad);
    event File(bytes32 indexed what, uint256 data);
    event File(bytes32 indexed what, address data);
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event DenyProxy(address indexed base, address indexed pause);
}
```

6.3.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "ESM/not-authorized");
        _;
    }
```

6.3.2 constructor(gem_, end_, proxy_, min_) X

```
    constructor(address gem_, address end_, address proxy_, uint256 min_) public
        ↪  {
        gem = GemLike(gem_);
        end = EndLike(end_);
        proxy = proxy_;
        min = min_;
        live = 1;

        wards[msg.sender] = 1;
        emit Rely(msg.sender);
    }
```

6.3.3 revokesGovernanceAccess()

```
    function revokesGovernanceAccess() external view returns (bool ret) {
        ret = proxy != address(0);
    }
```

6.3.4 add(x, y)

```
    // -- math --
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = x + y;
        require(z >= x);
    }
```

6.3.5 rely(usr) X a

```
    // --- Auth ---
    function rely(address usr) external auth {
        wards[usr] = 1;

        emit Rely(usr);
    }
```

6.3.6 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;

        emit Deny(usr);
    }
```

6.3.7 file(what, data) X a

```
    // -- admin --
    function file(bytes32 what, uint256 data) external auth {
        if (what == "min") {
            require(data > WAD, "ESM/min-too-small");
            min = data;
        } else {
            revert("ESM/file-unrecognized-param");
        }

        emit File(what, data);
    }
```

6.3.8 file(what, data) X a

```
    function file(bytes32 what, address data) external auth {
        if (what == "end") {
            end = EndLike(data);
        } else {
            revert("ESM/file-unrecognized-param");
        }

        emit File(what, data);
    }
```

6.3.9 cage() X a

```
    function cage() external auth {
        live = 0;
    }
```

6.3.10 fire() X

```
    function fire() external {
        require(live == 1, "ESM/permanently-disabled");
        require(Sum >= min,  "ESM/min-not-reached");

        if (proxy != address(0)) {
            DenyLike(end.vat()).deny(proxy);
        }
        end.cage();

        emit Fire();
    }
```

6.3.11 denyProxy(target) X

```
    function denyProxy(address target) external {
        require(live == 1, "ESM/permanently-disabled");
        require(Sum >= min,  "ESM/min-not-reached");

        DenyLike(target).deny(proxy);
        emit DenyProxy(target, proxy);
    }
```

6.3.12 join(wad) X

```
    function join(uint256 wad) external {
        require(live == 1, "ESM/permanently-disabled");
        require(end.live() == 1, "ESM/system-already-shutdown");

        sum[msg.sender] = add(sum[msg.sender], wad);
        Sum = add(Sum, wad);

        require(gem.transferFrom(msg.sender, address(this), wad), "ESM/transfer-
            ↪ failed");
        emit Join(msg.sender, wad);
    }
```

6.3.13 burn() X

```
    function burn() external {
        gem.burn(gem.balanceOf(address(this)));
    }
```

Chapter 7

Exec Lib

7.1 contract DssAction

```
abstract contract DssAction {

    using DssExecLib for *;

    // DssAction developer must override ‘actions()‘ and place all actions to be
        ↪ called inside.
    //   The DssExec function will call this subject to the officeHours limiter
    //   By keeping this function public we allow simulations of ‘execute()‘ on
        ↪ the actions outside of the cast time.
    function actions() public virtual;

    // Provides a descriptive tag for bot consumption
    // This should be modified weekly to provide a summary of the actions
    // Hash: seth keccak -- "$(wget https://<executive-vote-canonical-post> -q -
        ↪ O - 2>/dev/null)"
    function description() external view virtual returns (string memory);
}
```

7.1.1 modifier limited()

```
    // Modifier used to limit execution time when office hours is enabled
    modifier limited {
        require(DssExecLib.canCast(uint40(block.timestamp), officeHours()), "
            ↪ Outside office hours");
        _;
    }
```

7.1.2 officeHours()

```
    // Office Hours defaults to true by default.
    //   To disable office hours, override this function and
    //    return false in the inherited action.
    function officeHours() public view virtual returns (bool) {
        return true;
    }
```

7.1.3 execute() X

```
    // DssExec calls execute. We limit this function subject to officeHours
        ↪ modifier.
    function execute() external limited {
        actions();
    }
```

7.1.4 nextCastTime(eta)

```
    // Returns the next available cast time
    function nextCastTime(uint256 eta) external view returns (uint256 castTime)
       ↪ {
        require(eta <= type(uint40).max);
        castTime = DssExecLib.nextCastTime(uint40(eta), uint40(block.timestamp),
           ↪  officeHours());
    }
```

7.2 contract DssExec

```
contract DssExec {

    Changelog       constant public log   = Changelog(0
        ↪ xdA0Ab1e0017DEbCd72Be8599041a2aa3bA7e740F);
    uint256                  public eta;
    bytes                    public sig;
    bool                     public done;
    bytes32       immutable public tag;
    address       immutable public action;
    uint256       immutable public expiration;
    PauseAbstract immutable public pause;
}
```

7.2.1 description()

```
    // Provides a descriptive tag for bot consumption
    // This should be modified weekly to provide a summary of the actions
    // Hash: seth keccak -- "$(wget https://<executive-vote-canonical-post> -q -
        ↪ O - 2>/dev/null)"
    function description() external view returns (string memory) {
        return SpellAction(action).description();
    }
```

7.2.2 officeHours()

```
    function officeHours() external view returns (bool) {
        return SpellAction(action).officeHours();
    }
```

7.2.3 nextCastTime()

```
    function nextCastTime() external view returns (uint256 castTime) {
        return SpellAction(action).nextCastTime(eta);
    }
```

7.2.4 constructor(_expiration, _spellAction) X

```
    // @param _description  A string description of the spell
    // @param _expiration   The timestamp this spell will expire. (Ex. block.
        ↪ timestamp + 30 days)
    // @param _spellAction  The address of the spell action
    constructor(uint256 _expiration, address _spellAction) {
        pause       = PauseAbstract(log.getAddress("MCD_PAUSE"));
        expiration  = _expiration;
        action      = _spellAction;

        sig = abi.encodeWithSignature("execute()");
        bytes32 _tag;                    // Required for assembly access
        address _action = _spellAction;  // Required for assembly access
        assembly { _tag := extcodehash(_action) }
        tag = _tag;
    }
```

7.2.5 schedule() X

```
    function schedule() public {
        require(block.timestamp <= expiration, "This contract has expired");
        require(eta == 0, "This spell has already been scheduled");
        eta = block.timestamp + PauseAbstract(pause).delay();
        pause.plot(action, tag, sig, eta);
    }
```

7.2.6 cast() X

```
    function cast() public {
        require(!done, "spell-already-cast");
        done = true;
        pause.exec(action, tag, sig, eta);
    }
```

Chapter 8

Direct deposit

8.1 contract D3MAavePlan

```
contract D3MAavePlan is ID3MPlan {

    mapping (address => uint256) public wards;
    InterestRateStrategyLike     public tack;
    uint256                      public bar; // Target Interest Rate [ray]

    LendingPoolLike public immutable pool;
    TokenLike       public immutable stableDebt;
    TokenLike       public immutable variableDebt;
    TokenLike       public immutable dai;
    address         public immutable adai;
    uint256         public immutable adaiRevision;

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event File(bytes32 indexed what, uint256 data);
    event File(bytes32 indexed what, address data);

    // --- Math ---
    uint256 constant RAY = 10 ** 27;
}
```

8.1.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "D3MAavePlan/not-authorized");
        _;
    }
```

8.1.2 constructor(dai_, pool_) X

```
    constructor(address dai_, address pool_) {
        dai = TokenLike(dai_);
        pool = LendingPoolLike(pool_);

        // Fetch the reserve data from Aave
        (,,,,,,, address adai_, address stableDebt_, address variableDebt_,
           ↪ address interestStrategy_,) = pool.getReserveData(dai_);
        require(adai_             != address(0), "D3MAavePlan/invalid-adai");
        require(stableDebt_       != address(0), "D3MAavePlan/invalid-stableDebt
           ↪ ");
        require(variableDebt_     != address(0), "D3MAavePlan/invalid-
           ↪ variableDebt");
        require(interestStrategy_ != address(0), "D3MAavePlan/invalid-
           ↪ interestStrategy");
        adai         = adai_;
        adaiRevision = ATokenLike(adai_).ATOKEN_REVISION();
        stableDebt   = TokenLike(stableDebt_);
        variableDebt = TokenLike(variableDebt_);
        tack         = InterestRateStrategyLike(interestStrategy_);

        wards[msg.sender] = 1;
        emit Rely(msg.sender);
    }
```

8.1.3 _rmul(x, y)

```
    function _rmul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = (x * y) / RAY;
    }
```

8.1.4 _rdiv(x, y)

```
    function _rdiv(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = (x * RAY) / y;
    }
```

8.1.5 rely(usr) X a

```
    // --- Admin ---
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

8.1.6 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

8.1.7 file(what, data) X a

```
    function file(bytes32 what, uint256 data) external auth {
        if (what == "bar") bar = data;
        else revert("D3MAavePlan/file-unrecognized-param");
        emit File(what, data);
    }
```

8.1.8 file(what, data) X a

```
    function file(bytes32 what, address data) external auth {
        if (what == "tack") tack = InterestRateStrategyLike(data);
        else revert("D3MAavePlan/file-unrecognized-param");
        emit File(what, data);
    }
```

8.1.9 _calculateTargetSupply(targetInterestRate, totalDebt)

```
// --- Automated Rate targeting ---
function _calculateTargetSupply(uint256 targetInterestRate, uint256
   ↪ totalDebt) internal view returns (uint256) {
    uint256 base = tack.baseVariableBorrowRate();
    if (targetInterestRate <= base || targetInterestRate > tack.
        ↪ getMaxVariableBorrowRate()) {
        return 0;
    }

    // Do inverse calculation of interestStrategy
    uint256 variableRateSlope1 = tack.variableRateSlope1();

    uint256 targetUtil;
    if (targetInterestRate > base + variableRateSlope1) {
        // Excess interest rate
        uint256 r;
        unchecked {
            r = targetInterestRate - base - variableRateSlope1;
        }
        targetUtil = _rdiv(
                        _rmul(
                            tack.EXCESS_UTILIZATION_RATE(),
                            r
                        ),
                        tack.variableRateSlope2()
                     ) + tack.OPTIMAL_UTILIZATION_RATE();
    } else {
        // Optimal interest rate
        unchecked {
            targetUtil = _rdiv(
                            _rmul(
                                targetInterestRate - base,
                                tack.OPTIMAL_UTILIZATION_RATE()
                            ),
                            variableRateSlope1
                         );
        }
    }

    return _rdiv(totalDebt, targetUtil);
}
```

8.1.10 getTargetAssets(currentAssets)

```
// Note: This view function has no reentrancy protection.
//       On chain integrations should consider verifying `hub.locked()` is
   ↪ zero before relying on it.
function getTargetAssets(uint256 currentAssets) external override view
   ↪ returns (uint256) {
    uint256 targetInterestRate = bar;
    if (targetInterestRate == 0) return 0; // De-activated

    uint256 totalDebt = stableDebt.totalSupply() + variableDebt.totalSupply
        ↪ ();
    uint256 totalPoolSize = dai.balanceOf(adai) + totalDebt;
    uint256 targetTotalPoolSize = _calculateTargetSupply(targetInterestRate,
        ↪  totalDebt);

    if (targetTotalPoolSize >= totalPoolSize) {
        // Increase debt (or same)
        return currentAssets + (targetTotalPoolSize - totalPoolSize);
    } else {
        // Decrease debt
        unchecked {
            uint256 decrease = totalPoolSize - targetTotalPoolSize;
```

```
                if (currentAssets >= decrease) {
                    return currentAssets - decrease;
                } else {
                    return 0;
                }
            }
        }
    }
```

8.1.11 active()

```
    function active() public view override returns (bool) {
        if (bar == 0) return false;
        (,,,,,,, address adai_, address stableDebt_, address variableDebt_,
          ↪ address strategy,) = pool.getReserveData(address(dai));
        uint256 adaiRevision_ = ATokenLike(adai_).ATOKEN_REVISION();
        return strategy       == address(tack)          &&
               adai_          == address(adai)          &&
               adaiRevision_  == adaiRevision           &&
               stableDebt_    == address(stableDebt)    &&
               variableDebt_  == address(variableDebt);
    }
```

8.1.12 disable() X

```
    function disable() external override {
        require(wards[msg.sender] == 1 || !active(), "D3MAavePlan/not-authorized
          ↪ ");
        bar = 0; // ensure deactivation even if active conditions return later
        emit Disable();
    }
```

8.2 contract D3MCompoundPlan

```
contract D3MCompoundPlan is ID3MPlan {

    mapping (address => uint256) public wards;
    mapping (address => uint256) public tacks;     // supported rate models
    mapping (address => uint256) public delegates; // cDai supported
        ↪ implementations
    uint256                      public barb;      // target Interest Rate Per
        ↪ Block [wad] (0)

    CErc20Like public immutable cDai;

    // https://github.com/compound-finance/compound-protocol/blob/
        ↪ a3214f67b73310d547e00fc578e8355911c9d376/contracts/CTokenInterfaces.
        ↪ sol#L31
    uint256 internal constant MAX_BORROW_RATE = 0.0005e16;

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event File(bytes32 indexed what, uint256 data);
    event File(bytes32 indexed what, address addr, uint256 data);

    // --- Math ---
    uint256 internal constant WAD = 10 ** 18;
}
```

8.2.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "D3MCompoundPlan/not-authorized");
        _;
    }
```

8.2.2 constructor(cDai_) X

```
    constructor(address cDai_) {
        cDai = CErc20Like(cDai_);

        address rateModel_ = cDai.interestRateModel();
        address delegate_  = cDai.implementation();

        require(rateModel_ != address(0), "D3MCompoundPlan/invalid-rateModel");
        require(delegate_  != address(0), "D3MCompoundPlan/invalid-delegate");

        tacks[rateModel_]    = 1;
        delegates[delegate_] = 1;

        wards[msg.sender] = 1;
        emit Rely(msg.sender);
    }
```

8.2.3 _wmul(x, y)

```
    function _wmul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = (x * y) / WAD;
    }
```

8.2.4 _wdiv(x, y)

```
    function _wdiv(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = (x * WAD) / y;
    }
```

8.2.5 rely(usr) X a

```
    // --- Admin ---
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

8.2.6 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

8.2.7 file(what, data) X a

```
    function file(bytes32 what, uint256 data) external auth {
        if (what == "barb") {
            require(data <= MAX_BORROW_RATE, "D3MCompoundPlan/barb-too-high");
            barb = data;
        } else revert("D3MCompoundPlan/file-unrecognized-param");
        emit File(what, data);
    }
```

8.2.8 file(what, addr, data) X a

```
    function file(bytes32 what, address addr, uint256 data) external auth {
        require(data == 0 || data == 1, "D3MCompoundPlan/file-invalid-data");
        if (what == "tack") tacks[addr] = data;
        else if (what == "delegate") delegates[addr] = data;
        else revert("D3MCompoundPlan/file-unrecognized-param");
        emit File(what, addr, data);
    }
```

8.2.9 _calculateTargetSupply(targetInterestRate, borrows)

```
    function _calculateTargetSupply(uint256 targetInterestRate, uint256 borrows)
       ↪  internal view returns (uint256) {
        InterestRateModelLike tack = InterestRateModelLike(cDai.
           ↪ interestRateModel());
        require(tacks[address(tack)] == 1, "D3MCompoundPlan/invalid-tack");

        uint256 kink                   = tack.kink();
        uint256 multiplierPerBlock     = tack.multiplierPerBlock();
        uint256 baseRatePerBlock       = tack.baseRatePerBlock();
        uint256 jumpMultiplierPerBlock = tack.jumpMultiplierPerBlock();

        // The normal rate is a Compound term for the rate at kink utillization
        uint256 normalRate = _wmul(kink, multiplierPerBlock) + baseRatePerBlock;

        uint256 targetUtil;
        if (targetInterestRate > normalRate) {
            if (jumpMultiplierPerBlock == 0) return 0; // illegal rate, max is
               ↪ normal rate for this case
            targetUtil = kink + _wdiv(targetInterestRate - normalRate,
                ↪ jumpMultiplierPerBlock); // (1)
        } else if (targetInterestRate > baseRatePerBlock) {
            // multiplierPerBlock != 0, as otherwise normalRate ==
                ↪ baseRatePerBlock, matching the first case
            targetUtil = _wdiv(targetInterestRate - baseRatePerBlock,
                ↪ multiplierPerBlock);      // (2)
        } else {
            // if (target == base) => (borrows == 0) => supply does not matter
            // if (target  < base) => illegal rate
            return 0;
        }

        return _wdiv(borrows, targetUtil);
            ↪                                                   // (3)
    }
```

8.2.10 getTargetAssets(currentAssets)

```
    // Note: This view function has no reentrancy protection.
    //       On chain integrations should consider verifying ‘hub.locked()‘ is
        ↪ zero before relying on it.
    function getTargetAssets(uint256 currentAssets) external override view
        ↪ returns (uint256) {
        uint256 targetInterestRate = barb;
        if (targetInterestRate == 0) return 0; // De-activated

        uint256 borrows = cDai.totalBorrows();
        uint256 targetTotalPoolSize = _calculateTargetSupply(targetInterestRate,
            ↪  borrows);
        uint256 totalPoolSize = cDai.getCash() + borrows - cDai.totalReserves();

        if (targetTotalPoolSize >= totalPoolSize) {
            // Increase debt (or same)
            return currentAssets + (targetTotalPoolSize - totalPoolSize);
        } else {
            // Decrease debt
            unchecked {
                uint256 decrease = totalPoolSize - targetTotalPoolSize;
                if (currentAssets >= decrease) {
                    return currentAssets - decrease;
                } else {
                    return 0;
                }
            }
        }
    }
```

8.2.11 active()

```
    function active() public view override returns (bool) {
        if (barb == 0) return false;
        return tacks[cDai.interestRateModel()] == 1 &&
               delegates[cDai.implementation()] == 1;
    }
```

8.2.12 disable() X

```
    function disable() external override {
        require(wards[msg.sender] == 1 || !active(), "D3MCompoundPlan/not-
            ↪ authorized");
        barb = 0; // ensure deactivation even if active conditions return later
        emit Disable();
    }
```

8.3 contract D3MHub

```
/**
    @title D3M Hub
    @notice This is the main D3M contract and is responsible for winding and
    unwinding pools, interacting with DSS and tracking the plans and pools and
    their states.
*/
contract D3MHub {

    // --- Auth ---
    /**
        @notice Maps address that have permission in the Pool.
        @dev 1 = allowed, 0 = no permission
        @return authorization 1 or 0
    */
    mapping (address => uint256) public wards;

    address public vow;
    EndLike public end;
    uint256 public locked;

    /// @notice maps ilk bytes32 to the D3M tracking struct.
    mapping (bytes32 => Ilk) public ilks;

    VatLike     public immutable vat;
    DaiJoinLike public immutable daiJoin;

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event File(bytes32 indexed what, address data);
    event File(bytes32 indexed ilk, bytes32 indexed what, address data);
    event File(bytes32 indexed ilk, bytes32 indexed what, uint256 data);
    event Wind(bytes32 indexed ilk, uint256 amt);
    event Unwind(bytes32 indexed ilk, uint256 amt);
    event NoOp(bytes32 indexed ilk);
    event Fees(bytes32 indexed ilk, uint256 amt);
    event Exit(bytes32 indexed ilk, address indexed usr, uint256 amt);
    event Cage(bytes32 indexed ilk);
    event Cull(bytes32 indexed ilk, uint256 ink, uint256 art);
    event Uncull(bytes32 indexed ilk, uint256 wad);

    // --- Math ---
    uint256 internal constant WAD = 10 ** 18;
    uint256 internal constant RAY = 10 ** 27;
    uint256 internal constant MAXINT256 = uint256(type(int256).max);
    uint256 internal constant SAFEMAX = MAXINT256 / RAY;

    // --- Administration ---

    // Ilk Getters
}
```

8.3.1 struct D3MHub.Ilk

```
    /**
        @notice Tracking struct for each of the D3M ilks.
        @param pool   Contract to access external pool and hold balances
        @param plan   Contract used to calculate target debt
        @param tau    Time until you can write off the debt [sec]
        @param culled Debt write off triggered (1 or 0)
        @param tic    Timestamp when the pool is caged
    */
    struct Ilk {
        ID3MPool pool;   // Access external pool and holds balances
        ID3MPlan plan;   // How we calculate target debt
        uint256  tau;    // Time until you can write off the debt [sec]
        uint256  culled; // Debt write off triggered
        uint256  tic;    // Timestamp when the d3m can be culled (tau +
            ↪ timestamp when caged)
    }
```

8.3.2 modifier auth()

```
    /// @notice Modifier will revoke if msg.sender is not authorized.
    modifier auth {
        require(wards[msg.sender] == 1, "D3MHub/not-authorized");
        _;
    }
```

8.3.3 modifier lock()

```
    /// @notice Mutex to prevent reentrancy on external functions
    modifier lock {
        require(locked == 0, "D3MHub/system-locked");
        locked = 1;
        _;
        locked = 0;
    }
```

8.3.4 constructor(daiJoin_) X

```
    /**
        @dev sets msg.sender as authed.
        @param daiJoin_ address of the DSS Dai Join contract
    */
    constructor(address daiJoin_) {
        daiJoin = DaiJoinLike(daiJoin_);
        vat = VatLike(daiJoin.vat());
        TokenLike(daiJoin.dai()).approve(daiJoin_, type(uint256).max);
        vat.hope(daiJoin_);

        wards[msg.sender] = 1;
        emit Rely(msg.sender);
    }
```

8.3.5 _min(x, y)

```
    function _min(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = x <= y ? x : y;
    }
```

8.3.6 _max(x, y)

```
    function _max(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = x >= y ? x : y;
    }
```

8.3.7 _divup(x, y)

```
    function _divup(uint256 x, uint256 y) internal pure returns (uint256 z) {
        unchecked {
            z = x != 0 ? ((x - 1) / y) + 1 : 0;
        }
    }
```

8.3.8 rely(usr) X a

```
/**
    @notice Makes an address authorized to perform auth'ed functions.
    @dev msg.sender must be authorized.
    @param usr address to be authorized
*/
function rely(address usr) external auth {
    wards[usr] = 1;
    emit Rely(usr);
}
```

8.3.9 deny(usr) X a

```
/**
    @notice De-authorizes an address from performing auth'ed functions.
    @dev msg.sender must be authorized.
    @param usr address to be de-authorized
*/
function deny(address usr) external auth {
    wards[usr] = 0;
    emit Deny(usr);
}
```

8.3.10 file(what, data) X a

```
/**
    @notice update vow or end addresses.
    @dev msg.sender must be authorized.
    @param what name of what we are updating bytes32("vow"|"end")
    @param data address we are setting it to
*/
function file(bytes32 what, address data) external auth {
    require(vat.live() == 1, "D3MHub/no-file-during-shutdown");

    if (what == "vow") vow = data;
    else if (what == "end") end = EndLike(data);
    else revert("D3MHub/file-unrecognized-param");
    emit File(what, data);
}
```

8.3.11 file(ilk, what, data) X a

```
/**
    @notice update tau value for D3M ilk.
    @dev msg.sender must be authorized.
    @param ilk  bytes32 of the D3M ilk to be updated
    @param what bytes32("tau") or it will revert
    @param data number of seconds to wait after caging a pool to write off
       ↪ debt
*/
function file(bytes32 ilk, bytes32 what, uint256 data) external auth {
    if (what == "tau") ilks[ilk].tau = data;
    else revert("D3MHub/file-unrecognized-param");

    emit File(ilk, what, data);
}
```

8.3.12 file(ilk, what, data) X a

```
    /**
        @notice update plan or pool addresses for D3M ilk.
        @dev msg.sender must be authorized.
        @param ilk  bytes32 of the D3M ilk to be updated
        @param what bytes32("pool"|"plan") or it will revert
        @param data address we are setting it to
    */
    function file(bytes32 ilk, bytes32 what, address data) external auth {
        require(vat.live() == 1, "D3MHub/no-file-during-shutdown");
        require(ilks[ilk].tic == 0, "D3MHub/pool-not-live");

        if (what == "pool") ilks[ilk].pool = ID3MPool(data);
        else if (what == "plan") ilks[ilk].plan = ID3MPlan(data);
        else revert("D3MHub/file-unrecognized-param");
        emit File(ilk, what, data);
    }
```

8.3.13 _wipe(ilk, _pool, urn)

```
    // --- Internal functions that are called from exec(bytes32 ilk) ---

    function _wipe(bytes32 ilk, ID3MPool _pool, address urn) internal {
        uint256 amount = _pool.maxWithdraw();
        if (amount > 0) {
            _pool.withdraw(amount);
            daiJoin.join(address(this), amount);
            vat.move(address(this), vow, amount * RAY);

            uint256 toSlip = _min(vat.gem(ilk, urn), amount);
            // amount bounds toSlip and amount * RAY bounds amount to be much
                ↪ less than MAXINT256
            vat.slip(ilk, urn, -int256(toSlip));
            emit Unwind(ilk, amount);
        } else {
            emit NoOp(ilk);
        }
    }
```

8.3.14 _exec(ilk, _pool, Art, lineWad)

```
    function _exec(bytes32 ilk, ID3MPool _pool, uint256 Art, uint256 lineWad)
        ↪ internal {
        require(lineWad <= SAFEMAX, "D3MHub/lineWad-above-max-safe");
        (uint256 ink, uint256 art) = vat.urns(ilk, address(_pool));
        require(ink <= SAFEMAX, "D3MHub/ink-above-max-safe");
        require(ink >= art, "D3MHub/ink-not-greater-equal-art");
        require(art == Art, "D3MHub/more-than-one-urn");
        uint256 currentAssets = _pool.assetBalance(); // Should return DAI owned
            ↪ by D3MPool
        uint256 maxWithdraw = _min(_pool.maxWithdraw(), SAFEMAX);

        // Determine if fees were generated and try to account them (or the most
            ↪ that it is possible)
        if (currentAssets > ink) {
            uint256 fixInk = _min(
                _min(
                    currentAssets - ink, // fees generated
                    ink < lineWad // if previously CDP was under debt ceiling
                        ? (lineWad - ink) + maxWithdraw // up to gap to reach
                            ↪ debt ceiling + maxWithdraw
                        : maxWithdraw // up to maxWithdraw
                ),
                SAFEMAX + art - ink //  ensures that fixArt * RAY (rate) will be
                    ↪ <= MAXINT256 (in vat.grab)
            );
            vat.slip(ilk, address(_pool), int256(fixInk)); // Generate extra
                ↪ collateral
            vat.frob(ilk, address(_pool), address(_pool), address(this), int256(
                ↪ fixInk), 0); // Lock it
            unchecked {
                ink += fixInk; // can not overflow as worst case will be the
                    ↪ value of currentAssets
            }
            emit Fees(ilk, fixInk);
        }
        // Get the DAI and send as surplus (if there was permissionless DAI paid
            ↪ or fees accounted)
        if (art < ink) {
            address _vow = vow;
            uint256 fixArt;
            unchecked {
                fixArt = ink - art; // Amount of fees + permissionless DAI paid
                    ↪ we will now transform to debt
            }
            art = ink;
            vat.suck(_vow, _vow, fixArt * RAY); // This needs to be done to make
                ↪ sure we can deduct sin[vow] and vice in the next call
            // No need for ‘fixArt <= MAXINT256‘ require as:
            // MAXINT256 >>> MAXUINT256 / RAY which is already restricted above
            // Also fixArt should be always <= SAFEMAX (MAXINT256 / RAY)
            vat.grab(ilk, address(_pool), address(_pool), _vow, 0, int256(fixArt
                ↪ )); // Generating the debt
        }

        // Determine if it needs to unwind or wind
        uint256 toUnwind;
        uint256 toWind;

        // Determine if it needs to fully unwind due to D3M ilk being caged (but
            ↪ not culled), plan is not active or something
        // wrong is going with the third party and we are entering in the ilegal
            ↪ situation of having less assets than registered
        // It's adding up ‘WAD‘ due possible rounding errors
        if (ilks[ilk].tic != 0 || !ilks[ilk].plan.active() || currentAssets +
            ↪ WAD < ink) {
            toUnwind = maxWithdraw;
        } else {
            uint256 Line = vat.Line();
            uint256 debt = vat.debt();
            uint256 targetAssets = ilks[ilk].plan.getTargetAssets(currentAssets)
                ↪ ;

            // Determine if it needs to unwind due to:
            unchecked {
                toUnwind = _max(
                                _max(
                                    art > lineWad ? art - lineWad : 0, // ilk
                                        ↪ debt ceiling exceeded
                                    debt > Line ? _divup(debt - Line, RAY) : 0
                                        ↪ // global debt ceiling exceeded
                                ),
                                targetAssets < currentAssets ? currentAssets -
                                    ↪ targetAssets : 0 // plan targetAssets
                            );
                if (toUnwind > 0) {
                    toUnwind = _min(toUnwind, maxWithdraw);
                } else {
                    // Determine up to which value to wind:
                    // subtractions are safe as otherwise toUnwind > 0
                        ↪ conditional would be true
                    toWind = _min(
                                _min(
                                    _min(
                                        lineWad - art, // amount to reach ilk
                                            ↪ debt ceiling
                                        (Line - debt) / RAY  // amount to reach
                                            ↪ global debt ceiling
                                    ),
                                    targetAssets - currentAssets // plan
                                        ↪ targetAssets
                                ),
                                _pool.maxDeposit() // restricts winding if the
                                    ↪ pool has a max deposit
                            );
                }
            }
        }

        if (toUnwind > 0) {
            _pool.withdraw(toUnwind);
            daiJoin.join(address(this), toUnwind);
            // SAFEMAX bounds toUnwind making sure is <<< than MAXINT256
            vat.frob(ilk, address(_pool), address(_pool), address(this), -int256
                ↪ (toUnwind), -int256(toUnwind));
            vat.slip(ilk, address(_pool), -int256(toUnwind));
            emit Unwind(ilk, toUnwind);
        } else if (toWind > 0) {
            require(art + toWind <= SAFEMAX, "D3MHub/wind-overflow");
            vat.slip(ilk, address(_pool), int256(toWind));
            vat.frob(ilk, address(_pool), address(_pool), address(this), int256(
                ↪ toWind), int256(toWind));
            daiJoin.exit(address(_pool), toWind);
            _pool.deposit(toWind);
            emit Wind(ilk, toWind);
        } else {
            emit NoOp(ilk);
        }
    }
```

8.3.15 exec(ilk) X

```
    /**
        @notice Main function for updating a D3M position.
        Determines the current state and either winds or unwinds as necessary.
        @dev Winding the target position will be constrained by the Ilk debt
        ceiling, the overall DSS debt ceiling and the maximum deposit by the
        pool. Unwinding the target position will be constrained by the number
        of assets available to be withdrawn from the pool.
        @param ilk bytes32 of the D3M ilk name
    */
    function exec(bytes32 ilk) external lock {
        // IMPORTANT: this function assumes Vat rate of D3M ilks will always be
            ↪ == 1 * RAY (no fees).
        // That's why this module converts normalized debt (art) to Vat DAI
            ↪ generated with a simple RAY multiplication or division

        (uint256 Art, uint256 rate, uint256 spot, uint256 line,) = vat.ilks(ilk)
            ↪ ;
        require(rate == RAY, "D3MHub/rate-not-one");
        require(spot == RAY, "D3MHub/spot-not-one");

        ID3MPool _pool = ilks[ilk].pool;

        _pool.preDebtChange();

        if (vat.live() == 0) {
            // MCD caged
            // The main reason to have this case is trying to unwind the highest
                ↪  amount of DAI from the pool before end.debt is established.
            // That has the advantage to simplify End process, the best scenario
                ↪  would be unwinding everything which will decrease to the
            // minimum the amount of circulating supply of DAI, giving directly
                ↪ more value of other collaterals for each unit of DAI.
            // If this is not called, anyone can still call end.skim
                ↪ permissionlesly at any moment leaving remaining amount of
                ↪ pool shares
            // available to DAI holders to redeem it. This type of collateral is
                ↪  a cyclical one though, where user will need to go from
            // DAI -> pool share -> DAI -> ... making it not the most practical
                ↪ to handle. However, at the end, the net value of other
            // collaterals received per unit of DAI should end up being the same
                ↪  one (assuming there is liquidity in the pool to withdraw).
            EndLike _end = end;
            require(_end.debt() == 0, "D3MHub/end-debt-already-set");
            require(ilks[ilk].culled == 0, "D3MHub/module-has-to-be-unculled-
                ↪ first");
            _end.skim(ilk, address(_pool));
            _wipe(
                ilk,
                _pool,
                address(_end)
            );
        } else if (ilks[ilk].culled == 1) {
            _wipe(
                ilk,
                _pool,
                address(_pool)
            );
        } else {
            _exec(
                ilk,
                _pool,
                Art,
                line / RAY // round down ilk line in wad format
            );
        }

        _pool.postDebtChange();
    }
```

8.3.16 exit(ilk, usr, wad) X

```
    /**
        @notice Allow Users to return vat gem for Pool Shares.
        This will only occur during Global Settlement when users receive
        collateral for their Dai.
        @param ilk bytes32 of the D3M ilk name
        @param usr address that should receive the shares from the pool
        @param wad amount of gems that the msg.sender is returning
    */
    function exit(bytes32 ilk, address usr, uint256 wad) external lock {
        require(wad <= MAXINT256, "D3MHub/overflow");
        vat.slip(ilk, msg.sender, -int256(wad));
        ilks[ilk].pool.exit(usr, wad);
        emit Exit(ilk, usr, wad);
    }
```

8.3.17 cage(ilk) X a

```
    /**
        @notice Shutdown a pool.
        This starts the countdown to when the debt can be written off (cull).
        Once called, subsequent calls to `exec` will unwind as much of the
        position as possible.
        @dev msg.sender must be authorized.
        @param ilk bytes32 of the D3M ilk name
    */
    function cage(bytes32 ilk) external auth {
        require(vat.live() == 1, "D3MHub/no-cage-during-shutdown");
        require(ilks[ilk].tic == 0, "D3MHub/pool-already-caged");

        ilks[ilk].tic = block.timestamp + ilks[ilk].tau;
        emit Cage(ilk);
    }
```

8.3.18 cull(ilk) X

```
    /**
        @notice Write off the debt for a caged pool.
        This must occur while vat is live. Can be triggered by auth or
        after tau number of seconds has passed since the pool was caged.
        @dev This will send the pool's debt to the vow as sin and convert its
        collateral to gems.
        @param ilk bytes32 of the D3M ilk name
    */
    function cull(bytes32 ilk) external {
        require(vat.live() == 1, "D3MHub/no-cull-during-shutdown");

        uint256 _tic = ilks[ilk].tic;
        require(_tic > 0, "D3MHub/pool-live");

        require(_tic <= block.timestamp || wards[msg.sender] == 1, "D3MHub/
            ↪ unauthorized-cull");
        require(ilks[ilk].culled == 0, "D3MHub/already-culled");

        ID3MPool _pool = ilks[ilk].pool;

        (uint256 ink, uint256 art) = vat.urns(ilk, address(_pool));
        require(ink <= MAXINT256, "D3MHub/overflow");
        require(art <= MAXINT256, "D3MHub/overflow");
        vat.grab(ilk, address(_pool), address(_pool), vow, -int256(ink), -int256
            ↪ (art));

        ilks[ilk].culled = 1;
        emit Cull(ilk, ink, art);
    }
```

8.3.19 uncull(ilk) X

```
    /**
        @notice Rollback Write-off (cull) if General Shutdown happened.
        This function is required to have the collateral back in the vault so it
        can be taken by End module and eventually be shared to DAI holders (as
        any other collateral) or maybe even unwinded.
        @dev This pulls gems from the pool and reopens the urn with the gem
        amount of ink/art.
        @param ilk bytes32 of the D3M ilk name
    */
    function uncull(bytes32 ilk) external {
        ID3MPool _pool = ilks[ilk].pool;

        require(ilks[ilk].culled == 1, "D3MHub/not-prev-culled");
        require(vat.live() == 0, "D3MHub/no-uncull-normal-operation");

        address _vow = vow;
        uint256 wad = vat.gem(ilk, address(_pool));
        vat.suck(_vow, _vow, wad * RAY); // This needs to be done to make sure
            ↪ we can deduct sin[vow] and vice in the next call
        // wad * RAY bounds wad to be much less than MAXINT256
        vat.grab(ilk, address(_pool), address(_pool), _vow, int256(wad), int256(
            ↪ wad));
        ilks[ilk].culled = 0;
        emit Uncull(ilk, wad);
    }
```

8.3.20 pool(ilk)

```
    /**
        @notice Return pool of an ilk
        @param ilk   bytes32 of the D3M ilk
        @return pool address of pool contract
    */
    function pool(bytes32 ilk) external view returns (address) {
        return address(ilks[ilk].pool);
    }
```

8.3.21 plan(ilk)

```
    /**
        @notice Return plan of an ilk
        @param ilk   bytes32 of the D3M ilk
        @return plan address of plan contract
    */
    function plan(bytes32 ilk) external view returns (address) {
        return address(ilks[ilk].plan);
    }
```

8.3.22 tau(ilk)

```
    /**
        @notice Return tau of an ilk
        @param ilk  bytes32 of the D3M ilk
        @return tau sec until debt can be written off
    */
    function tau(bytes32 ilk) external view returns (uint256) {
        return ilks[ilk].tau;
    }
```

8.3.23 culled(ilk)

```
    /**
        @notice Return culled status of an ilk
        @param ilk  bytes32 of the D3M ilk
        @return culled whether or not the d3m has been culled
    */
    function culled(bytes32 ilk) external view returns (uint256) {
        return ilks[ilk].culled;
    }
```

8.3.24 tic(ilk)

```
    /**
        @notice Return tic of an ilk
        @param ilk  bytes32 of the D3M ilk
        @return tic timestamp of when d3m is caged
    */
    function tic(bytes32 ilk) external view returns (uint256) {
        return ilks[ilk].tic;
    }
```

8.4 contract D3MCoreDeployScript

```
contract D3MCoreDeployScript is Script {

    string constant NAME = "core";

    using stdJson for string;
    using ScriptTools for string;

    string config;
    DssInstance dss;

    address admin;
    D3MCoreInstance d3mCore;

}
```

Inherited:

```
// ?? SCRIPT
abstract contract Script is StdChains, StdCheatsSafe, StdUtils, ScriptBase {
    // Note: IS_SCRIPT() must return true.
    bool public IS_SCRIPT = true;
}
```

```
abstract contract ScriptBase is CommonBase {
    // Used when deploying with create2, https://github.com/Arachnid/
        ↪ deterministic-deployment-proxy.
    address internal constant CREATE2_FACTORY = 0
        ↪ x4e59b44847b379578588920cA78FbF26c0B4956C;

    VmSafe internal constant vmSafe = VmSafe(VM_ADDRESS);
}
```

```
abstract contract CommonBase {
    // Cheat code address, 0x7109709ECfa91a80626fF3989D68f67F5b1DD12D.
    address internal constant VM_ADDRESS = address(uint160(uint256(keccak256("
        ↪ hevm cheat code"))));
    // console.sol and console2.sol work by executing a staticcall to this
        ↪ address.
    address internal constant CONSOLE = 0
        ↪ x000000000000000000636F6e736F6c652e6c6f67;
    // Default address for tx.origin and msg.sender, 0
        ↪ x1804c8AB1F12E6bbf3894d4083f33e07309d1f38.
    address internal constant DEFAULT_SENDER = address(uint160(uint256(keccak256
        ↪ ("foundry default caller"))));
    // Address of the test contract, deployed by the DEFAULT_SENDER.
    address internal constant DEFAULT_TEST_CONTRACT = 0
        ↪ x5615dEB798BB3E4dFa0139dFa1b3D433Cc23b72f;
    // Deterministic deployment address of the Multicall3 contract.
    address internal constant MULTICALL3_ADDRESS = 0
        ↪ xcA11bde05977b3631167028862bE2a173976CA11;

    uint256 internal constant UINT256_MAX =
       115792089237316195423570985008687907853269984665640564039457584007913129639935
           ↪

    Vm internal constant vm = Vm(VM_ADDRESS);
    StdStorage internal stdstore;
}
```

```
abstract contract StdUtils {
    VmSafe private constant vm = VmSafe(address(uint160(uint256(keccak256("hevm
        ↪ cheat code")))));
    address private constant CONSOLE2_ADDRESS = 0
        ↪ x000000000000000000636F6e736F6c652e6c6f67;
    uint256 private constant INT256_MIN_ABS =
        57896044618658097711785492504343953926634992332820282019728792003956564819968;
        ↪
    uint256 private constant UINT256_MAX =
        115792089237316195423570985008687907853269984665640564039457584007913129639935;
        ↪
}
```

```
abstract contract StdCheatsSafe {
    Vm private constant vm = Vm(address(uint160(uint256(keccak256("hevm cheat
        ↪ code")))));

    bool private gasMeteringOff;
}
```

```
/**
 * StdChains provides information about EVM compatible chains that can be used
    ↪ in scripts/tests.
 * For each chain, the chain's name, chain ID, and a default RPC URL are
    ↪ provided. Chains are
 * identified by their alias, which is the same as the alias in the `[
    ↪ rpc_endpoints]` section of
 * the `foundry.toml` file. For best UX, ensure the alias in the `foundry.toml`
    ↪ file match the
 * alias used in this contract, which can be found as the first argument to the
 * `setChainWithDefaultRpcUrl` call in the `initialize` function.
 *
 * There are two main ways to use this contract:
 *   1. Set a chain with `setChain(string memory chainAlias, ChainData memory
    ↪ chain)` or
 *      `setChain(string memory chainAlias, Chain memory chain)`
 *   2. Get a chain with `getChain(string memory chainAlias)` or `getChain(
    ↪ uint256 chainId)`.
 *
 * The first time either of those are used, chains are initialized with the
    ↪ default set of RPC URLs.
 * This is done in `initialize`, which uses `setChainWithDefaultRpcUrl`.
    ↪ Defaults are recorded in
 * `defaultRpcUrls`.
 *
 * The `setChain` function is straightforward, and it simply saves off the given
    ↪  chain data.
 *
 * The `getChain` methods use `getChainWithUpdatedRpcUrl` to return a chain. For
    ↪  example, let's say
 * we want to retrieve `mainnet`'s RPC URL:
 *   - If you haven't set any mainnet chain info with `setChain`, you haven't
    ↪ specified that
 *     chain in `foundry.toml` and no env var is set, the default data and RPC
    ↪ URL will be returned.
 *   - If you have set a mainnet RPC URL in `foundry.toml` it will return that,
    ↪ if valid (e.g. if
 *     a URL is given or if an environment variable is given and that
    ↪ environment variable exists).
 *     Otherwise, the default data is returned.
 *   - If you specified data with `setChain` it will return that.
 *
 * Summarizing the above, the prioritization hierarchy is `setChain` -> `foundry
    ↪ .toml` -> environment variable -> defaults.
 */
abstract contract StdChains {
    VmSafe private constant vm = VmSafe(address(uint160(uint256(keccak256("hevm
        ↪ cheat code")))));

    bool private initialized;

    // Maps from the chain's alias (matching the alias in the `foundry.toml`
        ↪ file) to chain data.
    mapping(string => Chain) private chains;
    // Maps from the chain's alias to it's default RPC URL.
    mapping(string => string) private defaultRpcUrls;
    // Maps from a chain ID to it's alias.
    mapping(uint256 => string) private idToAlias;
}
```

8.4.1 struct StdChains.ChainData

```
    struct ChainData {
        string name;
        uint256 chainId;
        string rpcUrl;
    }
```

8.4.2 struct StdChains.Chain

```
    struct Chain {
        // The chain name.
        string name;
        // The chain's Chain ID.
        uint256 chainId;
        // The chain's alias. (i.e. what gets specified in `foundry.toml`).
        string chainAlias;
        // A default RPC endpoint for this chain.
        // NOTE: This default RPC URL is included for convenience to facilitate
            ↪ quick tests and
        // experimentation. Do not use this RPC URL for production test suites,
            ↪ CI, or other heavy
        // usage as you will be throttled and this is a disservice to others who
            ↪  need this endpoint.
        string rpcUrl;
    }
```

8.4.3 struct StdCheatsSafe.RawTx1559

```
    // Data structures to parse Transaction objects from the broadcast artifact
    // that conform to EIP1559. The Raw structs is what is parsed from the JSON
    // and then converted to the one that is used by the user for better UX.

    struct RawTx1559 {
        string[] arguments;
        address contractAddress;
        string contractName;
        // json value name = function
        string functionSig;
        bytes32 hash;
        // json value name = tx
        RawTx1559Detail txDetail;
        // json value name = type
        string opcode;
    }
```

8.4.4 struct StdCheatsSafe.RawTx1559Detail

```
    struct RawTx1559Detail {
        AccessList[] accessList;
        bytes data;
        address from;
        bytes gas;
        bytes nonce;
        address to;
        bytes txType;
        bytes value;
    }
```

8.4.5 struct StdCheatsSafe.Tx1559

```
    struct Tx1559 {
        string[] arguments;
        address contractAddress;
        string contractName;
        string functionSig;
        bytes32 hash;
        Tx1559Detail txDetail;
        string opcode;
    }
```

8.4.6 struct StdCheatsSafe.Tx1559Detail

```
    struct Tx1559Detail {
        AccessList[] accessList;
        bytes data;
        address from;
        uint256 gas;
        uint256 nonce;
        address to;
        uint256 txType;
        uint256 value;
    }
```

8.4.7 struct StdCheatsSafe.TxLegacy

```
    // Data structures to parse Transaction objects from the broadcast artifact
    // that DO NOT conform to EIP1559. The Raw structs is what is parsed from
       ↪ the JSON
    // and then converted to the one that is used by the user for better UX.

    struct TxLegacy {
        string[] arguments;
        address contractAddress;
        string contractName;
        string functionSig;
        string hash;
        string opcode;
        TxDetailLegacy transaction;
    }
```

8.4.8 struct StdCheatsSafe.TxDetailLegacy

```
    struct TxDetailLegacy {
        AccessList[] accessList;
        uint256 chainId;
        bytes data;
        address from;
        uint256 gas;
        uint256 gasPrice;
        bytes32 hash;
        uint256 nonce;
        bytes1 opcode;
        bytes32 r;
        bytes32 s;
        uint256 txType;
        address to;
        uint8 v;
        uint256 value;
    }
```

8.4.9 struct StdCheatsSafe.AccessList

```
    struct AccessList {
        address accessAddress;
        bytes32[] storageKeys;
    }
```

8.4.10 struct StdCheatsSafe.RawReceipt

```
    // Data structures to parse Receipt objects from the broadcast artifact.
    // The Raw structs is what is parsed from the JSON
    // and then converted to the one that is used by the user for better UX.

    struct RawReceipt {
        bytes32 blockHash;
        bytes blockNumber;
        address contractAddress;
        bytes cumulativeGasUsed;
        bytes effectiveGasPrice;
        address from;
        bytes gasUsed;
        RawReceiptLog[] logs;
        bytes logsBloom;
        bytes status;
        address to;
        bytes32 transactionHash;
        bytes transactionIndex;
    }
```

8.4.11 struct StdCheatsSafe.Receipt

```
    struct Receipt {
        bytes32 blockHash;
        uint256 blockNumber;
        address contractAddress;
        uint256 cumulativeGasUsed;
        uint256 effectiveGasPrice;
        address from;
        uint256 gasUsed;
        ReceiptLog[] logs;
        bytes logsBloom;
        uint256 status;
        address to;
        bytes32 transactionHash;
        uint256 transactionIndex;
    }
```

8.4.12 struct StdCheatsSafe.EIP1559ScriptArtifact

```
    // Data structures to parse the entire broadcast artifact, assuming the
    // transactions conform to EIP1559.

    struct EIP1559ScriptArtifact {
        string[] libraries;
        string path;
        string[] pending;
        Receipt[] receipts;
        uint256 timestamp;
        Tx1559[] transactions;
        TxReturn[] txReturns;
    }
```

8.4.13 struct StdCheatsSafe.RawEIP1559ScriptArtifact

```
    struct RawEIP1559ScriptArtifact {
        string[] libraries;
        string path;
        string[] pending;
        RawReceipt[] receipts;
        TxReturn[] txReturns;
        uint256 timestamp;
        RawTx1559[] transactions;
    }
```

8.4.14 struct StdCheatsSafe.RawReceiptLog

```
    struct RawReceiptLog {
        // json value = address
        address logAddress;
        bytes32 blockHash;
        bytes blockNumber;
        bytes data;
        bytes logIndex;
        bool removed;
        bytes32[] topics;
        bytes32 transactionHash;
        bytes transactionIndex;
        bytes transactionLogIndex;
    }
```

8.4.15 struct StdCheatsSafe.ReceiptLog

```
    struct ReceiptLog {
        // json value = address
        address logAddress;
        bytes32 blockHash;
        uint256 blockNumber;
        bytes data;
        uint256 logIndex;
        bytes32[] topics;
        uint256 transactionIndex;
        uint256 transactionLogIndex;
        bool removed;
    }
```

8.4.16 struct StdCheatsSafe.TxReturn

```
    struct TxReturn {
        string internalType;
        string value;
    }
```

8.4.17 modifier skipWhenForking() [StdCheatsSafe]

```
    modifier skipWhenForking() {
        if (!isFork()) {
            _;
        }
    }
```

8.4.18 modifier skipWhenNotForking() [StdCheatsSafe]

```
    modifier skipWhenNotForking() {
        if (isFork()) {
            _;
        }
    }
```

8.4.19 modifier noGasMetering() [StdCheatsSafe]

```
    modifier noGasMetering() {
        vm.pauseGasMetering();
        // To prevent turning gas monitoring back on with nested functions that
            ↪ use this modifier,
        // we check if gasMetering started in the off position. If it did, we
            ↪ don't want to turn
        // it back on until we exit the top level function that used the
            ↪ modifier
        //
        // i.e. funcA() noGasMetering { funcB() }, where funcB has noGasMetering
            ↪  as well.
        // funcA will have `gasStartedOff` as false, funcB will have it as true,
        // so we only turn metering back on at the end of the funcA
        bool gasStartedOff = gasMeteringOff;
        gasMeteringOff = true;

        _;

        // if gas metering was on when this modifier was called, turn it back on
            ↪  at the end
        if (!gasStartedOff) {
            gasMeteringOff = false;
            vm.resumeGasMetering();
        }
    }
```

8.4.20 _bound(x, min, max) [StdUtils]

```
    function _bound(uint256 x, uint256 min, uint256 max) internal pure virtual
        ↪ returns (uint256 result) {
        require(min <= max, "StdUtils bound(uint256,uint256,uint256): Max is
            ↪ less than min.");
        // If x is between min and max, return x directly. This is to ensure
            ↪ that dictionary values
        // do not get shifted if the min is nonzero. More info: https://github.
            ↪ com/foundry-rs/forge-std/issues/188
        if (x >= min && x <= max) return x;

        uint256 size = max - min + 1;

        // If the value is 0, 1, 2, 3, warp that to min, min+1, min+2, min+3.
            ↪ Similarly for the UINT256_MAX side.
        // This helps ensure coverage of the min/max values.
        if (x <= 3 && size > x) return min + x;
        if (x >= UINT256_MAX - 3 && size > UINT256_MAX - x) return max - (
            ↪ UINT256_MAX - x);

        // Otherwise, wrap x into the range [min, max], i.e. the range is
            ↪ inclusive.
        if (x > max) {
            uint256 diff = x - max;
            uint256 rem = diff % size;
            if (rem == 0) return max;
            result = min + rem - 1;
        } else if (x < min) {
            uint256 diff = min - x;
            uint256 rem = diff % size;
            if (rem == 0) return min;
            result = max - rem + 1;
        }
    }
```

8.4.21 bound(x, min, max) [StdUtils]

```
    function bound(uint256 x, uint256 min, uint256 max) internal view virtual
        ↪ returns (uint256 result) {
        result = _bound(x, min, max);
        console2_log("Bound Result", result);
    }
```

8.4.22 bound(x, min, max) [StdUtils]

```
    function bound(int256 x, int256 min, int256 max) internal view virtual
        ↪ returns (int256 result) {
        require(min <= max, "StdUtils bound(int256,int256,int256): Max is less
            ↪ than min.");

        // Shifting all int256 values to uint256 to use _bound function. The
            ↪ range of two types are:
        // int256 : -(2**255) ~ (2**255 - 1)
        // uint256:     0      ~ (2**256 - 1)
        // So, add 2**255, INT256_MIN_ABS to the integer values.
        //
        // If the given integer value is -2**255, we cannot use `-uint256(-x)`
            ↪ because of the overflow.
        // So, use `~uint256(x) + 1` instead.
        uint256 _x = x < 0 ? (INT256_MIN_ABS - ~uint256(x) - 1) : (uint256(x) +
            ↪ INT256_MIN_ABS);
        uint256 _min = min < 0 ? (INT256_MIN_ABS - ~uint256(min) - 1) : (uint256
            ↪ (min) + INT256_MIN_ABS);
        uint256 _max = max < 0 ? (INT256_MIN_ABS - ~uint256(max) - 1) : (uint256
            ↪ (max) + INT256_MIN_ABS);

        uint256 y = _bound(_x, _min, _max);

        // To move it back to int256 value, subtract INT256_MIN_ABS at here.
        result = y < INT256_MIN_ABS ? int256(~(INT256_MIN_ABS - y) + 1) : int256
            ↪ (y - INT256_MIN_ABS);
        console2_log("Bound result", vm.toString(result));
    }
```

8.4.23 computeCreateAddress(deployer, nonce) [StdUtils]

```
    /// @dev Compute the address a contract will be deployed at for a given
        ↪ deployer address and nonce
    /// @notice adapated from Solmate implementation (https://github.com/Rari-
        ↪ Capital/solmate/blob/main/src/utils/LibRLP.sol)
    function computeCreateAddress(address deployer, uint256 nonce) internal pure
        ↪  virtual returns (address) {
        // forgefmt: disable-start
        // The integer zero is treated as an empty byte string, and as a result
            ↪ it only has a length prefix, 0x80, computed via 0x80 + 0.
        // A one byte integer uses its own value as its length prefix, there is
            ↪ no additional "0x80 + length" prefix that comes before it.
        if (nonce == 0x00)      return addressFromLast20Bytes(keccak256(abi.
            ↪ encodePacked(bytes1(0xd6), bytes1(0x94), deployer, bytes1(0x80)))
            ↪ );
        if (nonce <= 0x7f)      return addressFromLast20Bytes(keccak256(abi.
            ↪ encodePacked(bytes1(0xd6), bytes1(0x94), deployer, uint8(nonce)))
            ↪ );
        // Nonces greater than 1 byte all follow a consistent encoding scheme,
           ↪ where each value is preceded by a prefix of 0x80 + length.
        if (nonce <= 2**8 - 1)  return addressFromLast20Bytes(keccak256(abi.
           ↪ encodePacked(bytes1(0xd7), bytes1(0x94), deployer, bytes1(0x81),
           ↪ uint8(nonce))));
        if (nonce <= 2**16 - 1) return addressFromLast20Bytes(keccak256(abi.
           ↪ encodePacked(bytes1(0xd8), bytes1(0x94), deployer, bytes1(0x82),
           ↪ uint16(nonce))));
        if (nonce <= 2**24 - 1) return addressFromLast20Bytes(keccak256(abi.
           ↪ encodePacked(bytes1(0xd9), bytes1(0x94), deployer, bytes1(0x83),
           ↪ uint24(nonce))));
        // forgefmt: disable-end

        // More details about RLP encoding can be found here: https://eth.wiki/
           ↪ fundamentals/rlp
        // 0xda = 0xc0 (short RLP prefix) + 0x16 (length of: 0x94 ++ proxy ++ 0
           ↪ x84 ++ nonce)
        // 0x94 = 0x80 + 0x14 (0x14 = the length of an address, 20 bytes, in hex
           ↪ )
        // 0x84 = 0x80 + 0x04 (0x04 = the bytes length of the nonce, 4 bytes, in
           ↪  hex)
        // We assume nobody can have a nonce large enough to require more than
           ↪ 32 bytes.
        return addressFromLast20Bytes(
            keccak256(abi.encodePacked(bytes1(0xda), bytes1(0x94), deployer,
               ↪ bytes1(0x84), uint32(nonce)))
        );
    }
```

8.4.24 computeCreate2Address(salt, initcodeHash, deployer) [StdUtils]

```
    function computeCreate2Address(bytes32 salt, bytes32 initcodeHash, address
        ↪ deployer)
        internal
        pure
        virtual
        returns (address)
    {
        return addressFromLast20Bytes(keccak256(abi.encodePacked(bytes1(0xff),
           ↪ deployer, salt, initcodeHash)));
    }
```

8.4.25 bytesToUint(b) [StdUtils]

```
    function bytesToUint(bytes memory b) internal pure virtual returns (uint256)
        ↪  {
        require(b.length <= 32, "StdUtils bytesToUint(bytes): Bytes length
           ↪ exceeds 32.");
        return abi.decode(abi.encodePacked(new bytes(32 - b.length), b), (
           ↪ uint256));
    }
```

8.4.26 addressFromLast20Bytes(bytesValue) [StdUtils]

```
    function addressFromLast20Bytes(bytes32 bytesValue) private pure returns (
        ↪ address) {
        return address(uint160(uint256(bytesValue)));
    }
```

8.4.27 console2_log(p0, p1) [StdUtils]

```
    // Used to prevent the compilation of console, which shortens the
       ↪ compilation time when console is not used elsewhere.

    function console2_log(string memory p0, uint256 p1) private view {
        (bool status,) = address(CONSOLE2_ADDRESS).staticcall(abi.
           ↪ encodeWithSignature("log(string,uint256)", p0, p1));
        status;
    }
```

8.4.28 console2_log(p0, p1) [StdUtils]

```
    function console2_log(string memory p0, string memory p1) private view {
        (bool status,) = address(CONSOLE2_ADDRESS).staticcall(abi.
           ↪ encodeWithSignature("log(string,string)", p0, p1));
        status;
    }
```

8.4.29 assumeNoPrecompiles(addr) [StdCheatsSafe]

```
    function assumeNoPrecompiles(address addr) internal virtual {
        // Assembly required since `block.chainid` was introduced in 0.8.0.
        uint256 chainId;
        assembly {
            chainId := chainid()
        }
        assumeNoPrecompiles(addr, chainId);
    }
```

8.4.30 assumeNoPrecompiles(addr, chainId) [StdCheatsSafe]

```
    function assumeNoPrecompiles(address addr, uint256 chainId) internal pure
       ↪ virtual {
        // Note: For some chains like Optimism these are technically predeploys
           ↪ (i.e. bytecode placed at a specific
        // address), but the same rationale for excluding them applies so we
           ↪ include those too.

        // These should be present on all EVM-compatible chains.
        vm.assume(addr < address(0x1) || addr > address(0x9));

        // forgefmt: disable-start
        if (chainId == 10 || chainId == 420) {
            // https://github.com/ethereum-optimism/optimism/blob/
               ↪ eaa371a0184b56b7ca6d9eb9cb0a2b78b2ccd864/op-bindings/
               ↪ predeploys/addresses.go#L6-L21
            vm.assume(addr < address(0x4200000000000000000000000000000000000000)
               ↪  || addr > address(0x4200000000000000000000000000000000000800
               ↪ ));
        } else if (chainId == 42161 || chainId == 421613) {
            // https://developer.arbitrum.io/useful-addresses#arbitrum-
               ↪ precompiles-l2-same-on-all-arb-chains
            vm.assume(addr < address(0x0000000000000000000000000000000000000064)
               ↪  || addr > address(0x0000000000000000000000000000000000000068
               ↪ ));
        } else if (chainId == 43114 || chainId == 43113) {
            // https://github.com/ava-labs/subnet-evm/blob/47
               ↪ c03fd007ecaa6de2c52ea081596e0a88401f58/precompile/params.go#
               ↪ L18-L59
            vm.assume(addr < address(0x0100000000000000000000000000000000000000)
               ↪  || addr > address(0x01000000000000000000000000000000000000ff
               ↪ ));
            vm.assume(addr < address(0x0200000000000000000000000000000000000000)
               ↪  || addr > address(0x02000000000000000000000000000000000000FF
               ↪ ));
            vm.assume(addr < address(0x0300000000000000000000000000000000000000)
                ↪ || addr > address(0x03000000000000000000000000000000000000Ff
                ↪ ));
        }
        // forgefmt: disable-end
    }
```

8.4.31 readEIP1559ScriptArtifact(path) [StdCheatsSafe]

```
    function readEIP1559ScriptArtifact(string memory path)
        internal
        view
        virtual
        returns (EIP1559ScriptArtifact memory)
    {
        string memory data = vm.readFile(path);
        bytes memory parsedData = vm.parseJson(data);
        RawEIP1559ScriptArtifact memory rawArtifact = abi.decode(parsedData, (
            ↪ RawEIP1559ScriptArtifact));
        EIP1559ScriptArtifact memory artifact;
        artifact.libraries = rawArtifact.libraries;
        artifact.path = rawArtifact.path;
        artifact.timestamp = rawArtifact.timestamp;
        artifact.pending = rawArtifact.pending;
        artifact.txReturns = rawArtifact.txReturns;
        artifact.receipts = rawToConvertedReceipts(rawArtifact.receipts);
        artifact.transactions = rawToConvertedEIPTx1559s(rawArtifact.
            ↪ transactions);
        return artifact;
    }
```

8.4.32 rawToConvertedEIPTx1559s(rawTxs) [StdCheatsSafe]

```
    function rawToConvertedEIPTx1559s(RawTx1559[] memory rawTxs) internal pure
        ↪ virtual returns (Tx1559[] memory) {
        Tx1559[] memory txs = new Tx1559[](rawTxs.length);
        for (uint256 i; i < rawTxs.length; i++) {
            txs[i] = rawToConvertedEIPTx1559(rawTxs[i]);
        }
        return txs;
    }
```

8.4.33 rawToConvertedEIPTx1559(rawTx) [StdCheatsSafe]

```
    function rawToConvertedEIPTx1559(RawTx1559 memory rawTx) internal pure
        ↪ virtual returns (Tx1559 memory) {
        Tx1559 memory transaction;
        transaction.arguments = rawTx.arguments;
        transaction.contractName = rawTx.contractName;
        transaction.functionSig = rawTx.functionSig;
        transaction.hash = rawTx.hash;
        transaction.txDetail = rawToConvertedEIP1559Detail(rawTx.txDetail);
        transaction.opcode = rawTx.opcode;
        return transaction;
    }
```

8.4.34 rawToConvertedEIP1559Detail(rawDetail) [StdCheatsSafe]

```
    function rawToConvertedEIP1559Detail(RawTx1559Detail memory rawDetail)
        internal
        pure
        virtual
        returns (Tx1559Detail memory)
    {
        Tx1559Detail memory txDetail;
        txDetail.data = rawDetail.data;
        txDetail.from = rawDetail.from;
        txDetail.to = rawDetail.to;
        txDetail.nonce = _bytesToUint(rawDetail.nonce);
        txDetail.txType = _bytesToUint(rawDetail.txType);
        txDetail.value = _bytesToUint(rawDetail.value);
        txDetail.gas = _bytesToUint(rawDetail.gas);
        txDetail.accessList = rawDetail.accessList;
        return txDetail;
    }
```

8.4.35 readTx1559s(path) [StdCheatsSafe]

```
    function readTx1559s(string memory path) internal view virtual returns (
        ↪ Tx1559[] memory) {
        string memory deployData = vm.readFile(path);
        bytes memory parsedDeployData = vm.parseJson(deployData, ".transactions"
            ↪ );
        RawTx1559[] memory rawTxs = abi.decode(parsedDeployData, (RawTx1559[]));
        return rawToConvertedEIPTx1559s(rawTxs);
    }
```

8.4.36 readTx1559(path, index) [StdCheatsSafe]

```
    function readTx1559(string memory path, uint256 index) internal view virtual
        ↪  returns (Tx1559 memory) {
        string memory deployData = vm.readFile(path);
        string memory key = string(abi.encodePacked(".transactions[", vm.
            ↪ toString(index), "]"));
        bytes memory parsedDeployData = vm.parseJson(deployData, key);
        RawTx1559 memory rawTx = abi.decode(parsedDeployData, (RawTx1559));
        return rawToConvertedEIPTx1559(rawTx);
    }
```

8.4.37 readReceipts(path) [StdCheatsSafe]

```
    // Analogous to readTransactions, but for receipts.
    function readReceipts(string memory path) internal view virtual returns (
        ↪ Receipt[] memory) {
        string memory deployData = vm.readFile(path);
        bytes memory parsedDeployData = vm.parseJson(deployData, ".receipts");
        RawReceipt[] memory rawReceipts = abi.decode(parsedDeployData, (
            ↪ RawReceipt[]));
        return rawToConvertedReceipts(rawReceipts);
    }
```

8.4.38 readReceipt(path, index) [StdCheatsSafe]

```
    function readReceipt(string memory path, uint256 index) internal view
        ↪ virtual returns (Receipt memory) {
        string memory deployData = vm.readFile(path);
        string memory key = string(abi.encodePacked(".receipts[", vm.toString(
            ↪ index), "]"));
        bytes memory parsedDeployData = vm.parseJson(deployData, key);
        RawReceipt memory rawReceipt = abi.decode(parsedDeployData, (RawReceipt)
            ↪ );
        return rawToConvertedReceipt(rawReceipt);
    }
```

8.4.39 rawToConvertedReceipts(rawReceipts) [StdCheatsSafe]

```
function rawToConvertedReceipts(RawReceipt[] memory rawReceipts) internal
    ↪ pure virtual returns (Receipt[] memory) {
    Receipt[] memory receipts = new Receipt[](rawReceipts.length);
    for (uint256 i; i < rawReceipts.length; i++) {
        receipts[i] = rawToConvertedReceipt(rawReceipts[i]);
    }
    return receipts;
}
```

8.4.40 rawToConvertedReceipt(rawReceipt) [StdCheatsSafe]

```
function rawToConvertedReceipt(RawReceipt memory rawReceipt) internal pure
    ↪ virtual returns (Receipt memory) {
    Receipt memory receipt;
    receipt.blockHash = rawReceipt.blockHash;
    receipt.to = rawReceipt.to;
    receipt.from = rawReceipt.from;
    receipt.contractAddress = rawReceipt.contractAddress;
    receipt.effectiveGasPrice = _bytesToUint(rawReceipt.effectiveGasPrice);
    receipt.cumulativeGasUsed = _bytesToUint(rawReceipt.cumulativeGasUsed);
    receipt.gasUsed = _bytesToUint(rawReceipt.gasUsed);
    receipt.status = _bytesToUint(rawReceipt.status);
    receipt.transactionIndex = _bytesToUint(rawReceipt.transactionIndex);
    receipt.blockNumber = _bytesToUint(rawReceipt.blockNumber);
    receipt.logs = rawToConvertedReceiptLogs(rawReceipt.logs);
    receipt.logsBloom = rawReceipt.logsBloom;
    receipt.transactionHash = rawReceipt.transactionHash;
    return receipt;
}
```

8.4.41 rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe]

```
function rawToConvertedReceiptLogs(RawReceiptLog[] memory rawLogs)
    internal
    pure
    virtual
    returns (ReceiptLog[] memory)
{
    ReceiptLog[] memory logs = new ReceiptLog[](rawLogs.length);
    for (uint256 i; i < rawLogs.length; i++) {
        logs[i].logAddress = rawLogs[i].logAddress;
        logs[i].blockHash = rawLogs[i].blockHash;
        logs[i].blockNumber = _bytesToUint(rawLogs[i].blockNumber);
        logs[i].data = rawLogs[i].data;
        logs[i].logIndex = _bytesToUint(rawLogs[i].logIndex);
        logs[i].topics = rawLogs[i].topics;
        logs[i].transactionIndex = _bytesToUint(rawLogs[i].transactionIndex)
            ↪ ;
        logs[i].transactionLogIndex = _bytesToUint(rawLogs[i].
            ↪ transactionLogIndex);
        logs[i].removed = rawLogs[i].removed;
    }
    return logs;
}
```

8.4.42 deployCode(what, args) [StdCheatsSafe]

```
// Deploy a contract by fetching the contract bytecode from
// the artifacts directory
// e.g. `deployCode(code, abi.encode(arg1,arg2,arg3))`
function deployCode(string memory what, bytes memory args) internal virtual
    ↪ returns (address addr) {
        bytes memory bytecode = abi.encodePacked(vm.getCode(what), args);
        /// @solidity memory-safe-assembly
        assembly {
            addr := create(0, add(bytecode, 0x20), mload(bytecode))
        }

        require(addr != address(0), "StdCheats deployCode(string,bytes):
            ↪ Deployment failed.");
    }
```

8.4.43 deployCode(what) [StdCheatsSafe]

```
    function deployCode(string memory what) internal virtual returns (address
        ↪ addr) {
        bytes memory bytecode = vm.getCode(what);
        /// @solidity memory-safe-assembly
        assembly {
            addr := create(0, add(bytecode, 0x20), mload(bytecode))
        }

        require(addr != address(0), "StdCheats deployCode(string): Deployment
            ↪ failed.");
    }
```

8.4.44 deployCode(what, args, val) [StdCheatsSafe]

```
    /// @dev deploy contract with value on construction
    function deployCode(string memory what, bytes memory args, uint256 val)
        ↪ internal virtual returns (address addr) {
        bytes memory bytecode = abi.encodePacked(vm.getCode(what), args);
        /// @solidity memory-safe-assembly
        assembly {
            addr := create(val, add(bytecode, 0x20), mload(bytecode))
        }

        require(addr != address(0), "StdCheats deployCode(string,bytes,uint256):
            ↪  Deployment failed.");
    }
```

8.4.45 deployCode(what, val) [StdCheatsSafe]

```
    function deployCode(string memory what, uint256 val) internal virtual
        ↪ returns (address addr) {
        bytes memory bytecode = vm.getCode(what);
        /// @solidity memory-safe-assembly
        assembly {
            addr := create(val, add(bytecode, 0x20), mload(bytecode))
        }

        require(addr != address(0), "StdCheats deployCode(string,uint256):
            ↪ Deployment failed.");
    }
```

8.4.46 makeAddrAndKey(name) [StdCheatsSafe]

```
    // creates a labeled address and the corresponding private key
    function makeAddrAndKey(string memory name) internal virtual returns (
        ↪ address addr, uint256 privateKey) {
        privateKey = uint256(keccak256(abi.encodePacked(name)));
        addr = vm.addr(privateKey);
        vm.label(addr, name);
    }
```

8.4.47 makeAddr(name) [StdCheatsSafe]

```
// creates a labeled address
function makeAddr(string memory name) internal virtual returns (address addr
    ↪ ) {
    (addr,) = makeAddrAndKey(name);
}
```

8.4.48 deriveRememberKey(mnemonic, index) [StdCheatsSafe]

```
function deriveRememberKey(string memory mnemonic, uint32 index)
    internal
    virtual
    returns (address who, uint256 privateKey)
{
    privateKey = vm.deriveKey(mnemonic, index);
    who = vm.rememberKey(privateKey);
}
```

8.4.49 _bytesToUint(b) [StdCheatsSafe]

```
function _bytesToUint(bytes memory b) private pure returns (uint256) {
    require(b.length <= 32, "StdCheats _bytesToUint(bytes): Bytes length
        ↪ exceeds 32.");
    return abi.decode(abi.encodePacked(new bytes(32 - b.length), b), (
        ↪ uint256));
}
```

8.4.50 isFork() [StdCheatsSafe]

```
function isFork() internal view virtual returns (bool status) {
    try vm.activeFork() {
        status = true;
    } catch (bytes memory) {}
}
```

8.4.51 getChain(chainAlias) [StdChains]

```
// The RPC URL will be fetched from config or defaultRpcUrls if possible.
function getChain(string memory chainAlias) internal virtual returns (Chain
    ↪ memory chain) {
    require(bytes(chainAlias).length != 0, "StdChains getChain(string):
        ↪ Chain alias cannot be the empty string.");

    initialize();
    chain = chains[chainAlias];
    require(
        chain.chainId != 0,
        string(abi.encodePacked("StdChains getChain(string): Chain with
            ↪ alias \"", chainAlias, "\" not found."))
    );

    chain = getChainWithUpdatedRpcUrl(chainAlias, chain);
}
```

8.4.52 getChain(chainId) [StdChains]

```
function getChain(uint256 chainId) internal virtual returns (Chain memory
    ↪ chain) {
    require(chainId != 0, "StdChains getChain(uint256): Chain ID cannot be
        ↪ 0.");
        initialize();
        string memory chainAlias = idToAlias[chainId];

        chain = chains[chainAlias];

        require(
            chain.chainId != 0,
            string(abi.encodePacked("StdChains getChain(uint256): Chain with ID
                ↪ ", vm.toString(chainId), " not found."))
        );

        chain = getChainWithUpdatedRpcUrl(chainAlias, chain);
    }
```

8.4.53 setChain(chainAlias, chain) [StdChains]

```
    // set chain info, with priority to argument's rpcUrl field.
    function setChain(string memory chainAlias, ChainData memory chain) internal
        ↪ virtual {
        require(
            bytes(chainAlias).length != 0,
            "StdChains setChain(string,ChainData): Chain alias cannot be the
                ↪ empty string."
        );

        require(chain.chainId != 0, "StdChains setChain(string,ChainData): Chain
            ↪ ID cannot be 0.");

        initialize();
        string memory foundAlias = idToAlias[chain.chainId];

        require(
            bytes(foundAlias).length == 0 || keccak256(bytes(foundAlias)) ==
                ↪ keccak256(bytes(chainAlias)),
            string(
                abi.encodePacked(
                    "StdChains setChain(string,ChainData): Chain ID ",
                    vm.toString(chain.chainId),
                    " already used by \"",
                    foundAlias,
                    "\"."
                )
            )
        );

        uint256 oldChainId = chains[chainAlias].chainId;
        delete idToAlias[oldChainId];

        chains[chainAlias] =
            Chain({name: chain.name, chainId: chain.chainId, chainAlias:
                ↪ chainAlias, rpcUrl: chain.rpcUrl});
        idToAlias[chain.chainId] = chainAlias;
    }
```

8.4.54 setChain(chainAlias, chain) [StdChains]

```
    // set chain info, with priority to argument's rpcUrl field.
    function setChain(string memory chainAlias, Chain memory chain) internal
        ↪ virtual {
        setChain(chainAlias, ChainData({name: chain.name, chainId: chain.chainId
            ↪ , rpcUrl: chain.rpcUrl}));
    }
```

8.4.55 _toUpper(str) [StdChains]

```
function _toUpper(string memory str) private pure returns (string memory) {
    bytes memory strb = bytes(str);
    bytes memory copy = new bytes(strb.length);
    for (uint256 i = 0; i < strb.length; i++) {
        bytes1 b = strb[i];
        if (b >= 0x61 && b <= 0x7A) {
            copy[i] = bytes1(uint8(b) - 32);
        } else {
            copy[i] = b;
        }
    }
    return string(copy);
}
```

8.4.56 getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains]

```
// lookup rpcUrl, in descending order of priority:
// current -> config (foundry.toml) -> environment variable -> default
function getChainWithUpdatedRpcUrl(string memory chainAlias, Chain memory
    ↪ chain) private returns (Chain memory) {
    if (bytes(chain.rpcUrl).length == 0) {
        try vm.rpcUrl(chainAlias) returns (string memory configRpcUrl) {
            chain.rpcUrl = configRpcUrl;
        } catch (bytes memory err) {
            chain.rpcUrl =
                vm.envOr(string(abi.encodePacked(_toUpper(chainAlias), "
                    ↪ _RPC_URL")), defaultRpcUrls[chainAlias]);
            // distinguish 'not found' from 'cannot read'
            bytes memory notFoundError =
                abi.encodeWithSignature("CheatCodeError", string(abi.
                    ↪ encodePacked("invalid rpc url ", chainAlias)));
            if (keccak256(notFoundError) != keccak256(err) || bytes(chain.
                ↪ rpcUrl).length == 0) {
                /// @solidity memory-safe-assembly
                assembly {
                    revert(add(32, err), mload(err))
                }
            }
        }
    }
    return chain;
}
```

8.4.57 initialize() [StdChains]

```
function initialize() private {
    if (initialized) return;

    initialized = true;

    // If adding an RPC here, make sure to test the default RPC URL in `
        ↪ testRpcs`
    setChainWithDefaultRpcUrl("anvil", ChainData("Anvil", 31337, "http
        ↪ ://127.0.0.1:8545"));
    setChainWithDefaultRpcUrl(
        "mainnet", ChainData("Mainnet", 1, "https://mainnet.infura.io/v3
            ↪ /6770454bc6ea42c58aac12978531b93f")
    );
    setChainWithDefaultRpcUrl(
        "goerli", ChainData("Goerli", 5, "https://goerli.infura.io/v3
            ↪ /6770454bc6ea42c58aac12978531b93f")
    );
    setChainWithDefaultRpcUrl(
        "sepolia", ChainData("Sepolia", 11155111, "https://sepolia.infura.io
            ↪ /v3/6770454bc6ea42c58aac12978531b93f")
    );
    setChainWithDefaultRpcUrl("optimism", ChainData("Optimism", 10, "https
        ↪ ://mainnet.optimism.io"));
    setChainWithDefaultRpcUrl("optimism_goerli", ChainData("Optimism Goerli"
        ↪ , 420, "https://goerli.optimism.io"));
    setChainWithDefaultRpcUrl("arbitrum_one", ChainData("Arbitrum One",
        ↪ 42161, "https://arb1.arbitrum.io/rpc"));
    setChainWithDefaultRpcUrl(
        "arbitrum_one_goerli", ChainData("Arbitrum One Goerli", 421613, "
            ↪ https://goerli-rollup.arbitrum.io/rpc")
    );
    setChainWithDefaultRpcUrl("arbitrum_nova", ChainData("Arbitrum Nova",
        ↪ 42170, "https://nova.arbitrum.io/rpc"));
    setChainWithDefaultRpcUrl("polygon", ChainData("Polygon", 137, "https://
        ↪ polygon-rpc.com"));
    setChainWithDefaultRpcUrl(
        "polygon_mumbai", ChainData("Polygon Mumbai", 80001, "https://rpc-
            ↪ mumbai.maticvigil.com")
    );
    setChainWithDefaultRpcUrl("avalanche", ChainData("Avalanche", 43114, "
        ↪ https://api.avax.network/ext/bc/C/rpc"));
    setChainWithDefaultRpcUrl(
        "avalanche_fuji", ChainData("Avalanche Fuji", 43113, "https://api.
            ↪ avax-test.network/ext/bc/C/rpc")
    );
    setChainWithDefaultRpcUrl(
        "bnb_smart_chain", ChainData("BNB Smart Chain", 56, "https://bsc-
            ↪ dataseed1.binance.org")
    );
    setChainWithDefaultRpcUrl(
        "bnb_smart_chain_testnet",
        ChainData("BNB Smart Chain Testnet", 97, "https://data-seed-prebsc
            ↪ -1-s1.binance.org:8545")
    );
    setChainWithDefaultRpcUrl("gnosis_chain", ChainData("Gnosis Chain", 100,
        ↪  "https://rpc.gnosischain.com"));
}
```

8.4.58 setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains]

```
    // set chain info, with priority to chainAlias' rpc url in foundry.toml
    function setChainWithDefaultRpcUrl(string memory chainAlias, ChainData
        ↪ memory chain) private {
        string memory rpcUrl = chain.rpcUrl;
        defaultRpcUrls[chainAlias] = rpcUrl;
        chain.rpcUrl = "";
        setChain(chainAlias, chain);
        chain.rpcUrl = rpcUrl; // restore argument
    }
```

8.4.59 run() X

```
    function run() external {
        config = ScriptTools.loadConfig(NAME);
        dss = MCD.loadFromChainlog(config.readAddress("chainlog"));

        admin = config.readAddress("admin");

        vm.startBroadcast();
        d3mCore = D3MDeploy.deployCore(
            msg.sender,
            admin,
            address(dss.daiJoin)
        );
        vm.stopBroadcast();

        ScriptTools.exportContract(NAME, "hub", d3mCore.hub);
        ScriptTools.exportContract(NAME, "mom", d3mCore.mom);
    }
```

8.5 contract D3MAavePool

```
contract D3MAavePool is ID3MPool {

    mapping (address => uint256) public wards;
    address                      public hub;
    address                      public king; // Who gets the rewards
    uint256                      public exited;

    bytes32          public immutable ilk;
    VatLike          public immutable vat;
    LendingPoolLike public immutable pool;
    ATokenLike       public immutable stableDebt;
    ATokenLike       public immutable variableDebt;
    ATokenLike       public immutable adai;
    TokenLike        public immutable dai; // Asset

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event File(bytes32 indexed what, address data);
    event Collect(address indexed king, address indexed gift, uint256 amt);

    // --- Math ---
    uint256 internal constant RAY = 10 ** 27;

    function preDebtChange() external override {}

    function postDebtChange() external override {}
}
```

8.5.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "D3MAavePool/not-authorized");
        _;
    }
```

8.5.2 modifier onlyHub()

```
    modifier onlyHub {
        require(msg.sender == hub, "D3MAavePool/only-hub");
        _;
    }
```

8.5.3 constructor(ilk_, hub_, dai_, pool_) X

```
    constructor(bytes32 ilk_, address hub_, address dai_, address pool_) {
        ilk = ilk_;
        dai = TokenLike(dai_);
        pool = LendingPoolLike(pool_);

        // Fetch the reserve data from Aave
        (,,,,,,, address adai_, address stableDebt_, address variableDebt_,,) =
            ↪ pool.getReserveData(dai_);
        require(adai_         != address(0), "D3MAavePool/invalid-adai");
        require(stableDebt_   != address(0), "D3MAavePool/invalid-stableDebt");
        require(variableDebt_ != address(0), "D3MAavePool/invalid-variableDebt")
            ↪ ;

        adai = ATokenLike(adai_);
        stableDebt = ATokenLike(stableDebt_);
        variableDebt = ATokenLike(variableDebt_);
        dai.approve(pool_, type(uint256).max);

        hub = hub_;
        vat = VatLike(D3mHubLike(hub_).vat());
        vat.hope(hub_);

        wards[msg.sender] = 1;
        emit Rely(msg.sender);
    }
```

8.5.4 _rdiv(x, y)

```
    function _rdiv(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = (x * RAY) / y;
    }
```

8.5.5 _min(x, y)

```
    function _min(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = x <= y ? x : y;
    }
```

8.5.6 rely(usr) X a

```
    // --- Admin ---
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

8.5.7 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

8.5.8 file(what, data) X a

```
    function file(bytes32 what, address data) external auth {
        require(vat.live() == 1, "D3MAavePool/no-file-during-shutdown");
        if (what == "hub") {
            vat.nope(hub);
            hub = data;
            vat.hope(data);
        } else if (what == "king") king = data;
        else revert("D3MAavePool/file-unrecognized-param");
        emit File(what, data);
    }
```

8.5.9 deposit(wad) X

```
    // Deposits Dai to Aave in exchange for adai which is received by this
       ↪ contract
    // Aave: https://docs.aave.com/developers/v/2.0/the-core-protocol/
       ↪ lendingpool#deposit
    function deposit(uint256 wad) external override onlyHub {
        uint256 scaledPrev = adai.scaledBalanceOf(address(this));
        pool.deposit(address(dai), wad, address(this), 0);

        // Verify the correct amount of adai shows up
        uint256 interestIndex = pool.getReserveNormalizedIncome(address(dai));
        uint256 scaledAmount = _rdiv(wad, interestIndex);
        require(adai.scaledBalanceOf(address(this)) >= (scaledPrev +
            ↪ scaledAmount), "D3MAavePool/incorrect-adai-balance-received");
    }
```

8.5.10 withdraw(wad) X

```
    // Withdraws Dai from Aave in exchange for adai
    // Aave: https://docs.aave.com/developers/v/2.0/the-core-protocol/
        ↪ lendingpool#withdraw
    function withdraw(uint256 wad) external override onlyHub {
        uint256 prevDai = dai.balanceOf(msg.sender);

        pool.withdraw(address(dai), wad, msg.sender);

        require(dai.balanceOf(msg.sender) == prevDai + wad, "D3MAavePool/
            ↪ incorrect-dai-balance-received");
    }
```

8.5.11 exit(dst, wad) X

```
    function exit(address dst, uint256 wad) external override onlyHub {
        uint256 exited_ = exited;
        exited = exited_ + wad;
        uint256 amt = wad * assetBalance() / (D3mHubLike(hub).end().Art(ilk) -
            ↪ exited_);
        require(adai.transfer(dst, amt), "D3MAavePool/transfer-failed");
    }
```

8.5.12 quit(dst) X a

```
    function quit(address dst) external override auth {
        require(vat.live() == 1, "D3MAavePool/no-quit-during-shutdown");
        require(adai.transfer(dst, adai.balanceOf(address(this))), "D3MAavePool/
            ↪ transfer-failed");
    }
```

8.5.13 assetBalance()

```
    // --- Balance of the underlying asset (Dai)
    function assetBalance() public view override returns (uint256) {
        return adai.balanceOf(address(this));
    }
```

8.5.14 maxDeposit()

```
    function maxDeposit() external pure override returns (uint256) {
        return type(uint256).max;
    }
```

8.5.15 maxWithdraw()

```
    function maxWithdraw() external view override returns (uint256) {
        return _min(dai.balanceOf(address(adai)), assetBalance());
    }
```

8.5.16 redeemable()

```
    function redeemable() external view override returns (address) {
        return address(adai);
    }
```

8.5.17 collect() X

```
    // --- Collect any rewards ---
    function collect() external returns (uint256 amt) {
        require(king != address(0), "D3MAavePool/king-not-set");

        address[] memory assets = new address[](1);
        assets[0] = address(adai);

        RewardsClaimerLike rewardsClaimer = RewardsClaimerLike(adai.
            ↪ getIncentivesController());

        amt = rewardsClaimer.claimRewards(assets, type(uint256).max, king);
        address gift = rewardsClaimer.REWARD_TOKEN();
        emit Collect(king, gift, amt);
    }
```

8.6 contract D3MOracle

```
contract D3MOracle {
    // --- Auth ---
    /**
        @notice Maps address that have permission in the Pool.
        @dev 1 = allowed, 0 = no permission
        @return authorization 1 or 0
    */
    mapping (address => uint256) public wards;
    address public hub;

    address public immutable vat;
    bytes32 public immutable ilk;

    uint256 internal constant WAD = 10 ** 18;

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event File(bytes32 indexed what, address data);

    // --- Administration ---
}
```

8.6.1 modifier auth()

```
    /// @notice Modifier will revoke if msg.sender is not authorized.
    modifier auth {
        require(wards[msg.sender] == 1, "D3MOracle/not-authorized");
        _;
    }
```

8.6.2 constructor(vat_, ilk_) X

```
    constructor(address vat_, bytes32 ilk_) {
        vat = vat_;
        ilk = ilk_;

        wards[msg.sender] = 1;
        emit Rely(msg.sender);
    }
```

8.6.3 rely(usr) X a

```
    /**
        @notice Makes an address authorized to perform auth'ed functions.
        @dev msg.sender must be authorized.
        @param usr address to be authorized
    */
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

8.6.4 deny(usr) X a

```
    /**
        @notice De-authorizes an address from performing auth'ed functions.
        @dev msg.sender must be authorized.
        @param usr address to be de-authorized
    */
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

8.6.5 file(what, data) X a

```
    /**
        @notice update an address.
        @dev msg.sender must be authorized.
        @param what name of what we are updating.
        @param data address we are setting it to
    */
    function file(bytes32 what, address data) external auth {
        require(VatLike(vat).live() == 1, "D3MOracle/no-file-during-shutdown");

        if (what == "hub") hub = data;
        else revert("D3MOracle/file-unrecognized-param");
        emit File(what, data);
    }
```

8.6.6 peek()

```
    /**
        @notice Return value and status of the oracle
        @return val always 1 WAD
        @return ok true if vat is live or ilk is not culled
    */
    function peek() public view returns (uint256 val, bool ok) {
        val = WAD;
        ok = VatLike(vat).live() == 1 || HubLike(hub).culled(ilk) == 0;
    }
```

8.6.7 read()

```
    /**
        @notice Return value
        @dev vat must be live or ilk must not be culled  in hub.
        @return val always 1 WAD value
    */
    function read() external view returns (uint256 val) {
        bool ok;
        (val, ok) = peek();
        require(ok, "D3MOracle/ilk-culled-in-shutdown"); // In order to stop end
            ↪ .cage(ilk) until is unculled
    }
```

8.7 contract D3MMom

```
// Bypass governance delay to disable a direct deposit module
contract D3MMom {
    address public owner;
    address public authority;

    event SetOwner(address indexed newOwner);
    event SetAuthority(address indexed newAuthority);
    event Disable(address indexed who);
}
```

8.7.1 modifier onlyOwner()

```
    modifier onlyOwner {
        require(msg.sender == owner, "D3MMom/only-owner");
        _;
    }
```

8.7.2 modifier auth()

```
    modifier auth {
        require(isAuthorized(msg.sender, msg.sig), "D3MMom/not-authorized");
        _;
    }
```

8.7.3 constructor() X

```
    constructor() {
        owner = msg.sender;
        emit SetOwner(msg.sender);
    }
```

8.7.4 isAuthorized(src, sig)

```
    function isAuthorized(address src, bytes4 sig) internal view returns (bool)
       ↪ {
        if (src == address(this)) {
            return true;
        } else if (src == owner) {
            return true;
        } else if (authority == address(0)) {
            return false;
        } else {
            return AuthorityLike(authority).canCall(src, address(this), sig);
        }
    }
```

8.7.5 setOwner(owner_) X

```
    // Governance actions with delay
    function setOwner(address owner_) external onlyOwner {
        owner = owner_;
        emit SetOwner(owner_);
    }
```

8.7.6 setAuthority(authority_) X

```
    function setAuthority(address authority_) external onlyOwner {
        authority = authority_;
        emit SetAuthority(authority_);
    }
```

8.7.7 disable(who) X a

```
    // Governance action without delay
    function disable(address who) external auth {
        DisableLike(who).disable();
        emit Disable(who);
    }
```

8.8 contract D3MCoreInitScript

```
contract D3MCoreInitScript is Script {

    string constant NAME = "core";

    using stdJson for string;
    using ScriptTools for string;

    string config;
    string dependencies;
    DssInstance dss;

    D3MCoreInstance d3mCore;

}
```

Inherited:

```
// ?? SCRIPT
abstract contract Script is StdChains, StdCheatsSafe, StdUtils, ScriptBase {
    // Note: IS_SCRIPT() must return true.
    bool public IS_SCRIPT = true;
}
```

```
abstract contract ScriptBase is CommonBase {
    // Used when deploying with create2, https://github.com/Arachnid/
        ↪ deterministic-deployment-proxy.
    address internal constant CREATE2_FACTORY = 0
        ↪ x4e59b44847b379578588920cA78FbF26c0B4956C;

    VmSafe internal constant vmSafe = VmSafe(VM_ADDRESS);
}
```

```
abstract contract CommonBase {
    // Cheat code address, 0x7109709ECfa91a80626fF3989D68f67F5b1DD12D.
    address internal constant VM_ADDRESS = address(uint160(uint256(keccak256("
        ↪ hevm cheat code"))));
    // console.sol and console2.sol work by executing a staticcall to this
        ↪ address.
    address internal constant CONSOLE = 0
        ↪ x000000000000000000636F6e736F6c652e6c6f67;
    // Default address for tx.origin and msg.sender, 0
        ↪ x1804c8AB1F12E6bbf3894d4083f33e07309d1f38.
    address internal constant DEFAULT_SENDER = address(uint160(uint256(keccak256
        ↪ ("foundry default caller"))));
    // Address of the test contract, deployed by the DEFAULT_SENDER.
    address internal constant DEFAULT_TEST_CONTRACT = 0
        ↪ x5615dEB798BB3E4dFa0139dFa1b3D433Cc23b72f;
    // Deterministic deployment address of the Multicall3 contract.
    address internal constant MULTICALL3_ADDRESS = 0
        ↪ xcA11bde05977b3631167028862bE2a173976CA11;

    uint256 internal constant UINT256_MAX =
        115792089237316195423570985008687907853269984665640564039457584007913129639935
            ↪

    Vm internal constant vm = Vm(VM_ADDRESS);
    StdStorage internal stdstore;
}
```

```
abstract contract StdUtils {
    VmSafe private constant vm = VmSafe(address(uint160(uint256(keccak256("hevm
        ↪ cheat code")))));
    address private constant CONSOLE2_ADDRESS = 0
        ↪ x000000000000000000636F6e736F6c652e6c6f67;
    uint256 private constant INT256_MIN_ABS =
        57896044618658097711785492504343953926634992332820282019728792003956564819968;
        ↪
    uint256 private constant UINT256_MAX =
        115792089237316195423570985008687907853269984665640564039457584007913129639935;
        ↪
}
```

```
abstract contract StdCheatsSafe {
    Vm private constant vm = Vm(address(uint160(uint256(keccak256("hevm cheat
        ↪ code")))));

    bool private gasMeteringOff;
}
```

```
/**
 * StdChains provides information about EVM compatible chains that can be used
    ↪ in scripts/tests.
 * For each chain, the chain's name, chain ID, and a default RPC URL are
    ↪ provided. Chains are
 * identified by their alias, which is the same as the alias in the `[
    ↪ rpc_endpoints]` section of
 * the `foundry.toml` file. For best UX, ensure the alias in the `foundry.toml`
    ↪ file match the
 * alias used in this contract, which can be found as the first argument to the
 * `setChainWithDefaultRpcUrl` call in the `initialize` function.
 *
 * There are two main ways to use this contract:
 *   1. Set a chain with `setChain(string memory chainAlias, ChainData memory
    ↪ chain)` or
 *      `setChain(string memory chainAlias, Chain memory chain)`
 *   2. Get a chain with `getChain(string memory chainAlias)` or `getChain(
    ↪ uint256 chainId)`.
 *
 * The first time either of those are used, chains are initialized with the
    ↪ default set of RPC URLs.
 * This is done in `initialize`, which uses `setChainWithDefaultRpcUrl`.
    ↪ Defaults are recorded in
 * `defaultRpcUrls`.
 *
 * The `setChain` function is straightforward, and it simply saves off the given
    ↪  chain data.
 *
 * The `getChain` methods use `getChainWithUpdatedRpcUrl` to return a chain. For
    ↪  example, let's say
 * we want to retrieve `mainnet`'s RPC URL:
 *   - If you haven't set any mainnet chain info with `setChain`, you haven't
    ↪ specified that
 *     chain in `foundry.toml` and no env var is set, the default data and RPC
    ↪ URL will be returned.
 *   - If you have set a mainnet RPC URL in `foundry.toml` it will return that,
    ↪ if valid (e.g. if
 *     a URL is given or if an environment variable is given and that
    ↪ environment variable exists).
 *     Otherwise, the default data is returned.
 *   - If you specified data with `setChain` it will return that.
 *
 * Summarizing the above, the prioritization hierarchy is `setChain` -> `foundry
    ↪ .toml` -> environment variable -> defaults.
 */
abstract contract StdChains {
    VmSafe private constant vm = VmSafe(address(uint160(uint256(keccak256("hevm
        ↪ cheat code")))));

    bool private initialized;

    // Maps from the chain's alias (matching the alias in the `foundry.toml`
        ↪ file) to chain data.
    mapping(string => Chain) private chains;
    // Maps from the chain's alias to it's default RPC URL.
    mapping(string => string) private defaultRpcUrls;
    // Maps from a chain ID to it's alias.
    mapping(uint256 => string) private idToAlias;
}
```

8.8.1 struct StdChains.ChainData

```
    struct ChainData {
        string name;
        uint256 chainId;
        string rpcUrl;
    }
```

8.8.2 struct StdChains.Chain

```
    struct Chain {
        // The chain name.
        string name;
        // The chain's Chain ID.
        uint256 chainId;
        // The chain's alias. (i.e. what gets specified in `foundry.toml`).
        string chainAlias;
        // A default RPC endpoint for this chain.
        // NOTE: This default RPC URL is included for convenience to facilitate
            ↪ quick tests and
        // experimentation. Do not use this RPC URL for production test suites,
            ↪ CI, or other heavy
        // usage as you will be throttled and this is a disservice to others who
            ↪  need this endpoint.
        string rpcUrl;
    }
```

8.8.3 struct StdCheatsSafe.RawTx1559

```
    // Data structures to parse Transaction objects from the broadcast artifact
    // that conform to EIP1559. The Raw structs is what is parsed from the JSON
    // and then converted to the one that is used by the user for better UX.

    struct RawTx1559 {
        string[] arguments;
        address contractAddress;
        string contractName;
        // json value name = function
        string functionSig;
        bytes32 hash;
        // json value name = tx
        RawTx1559Detail txDetail;
        // json value name = type
        string opcode;
    }
```

8.8.4 struct StdCheatsSafe.RawTx1559Detail

```
    struct RawTx1559Detail {
        AccessList[] accessList;
        bytes data;
        address from;
        bytes gas;
        bytes nonce;
        address to;
        bytes txType;
        bytes value;
    }
```

8.8.5 struct StdCheatsSafe.Tx1559

```
    struct Tx1559 {
        string[] arguments;
        address contractAddress;
        string contractName;
        string functionSig;
        bytes32 hash;
        Tx1559Detail txDetail;
        string opcode;
    }
```

8.8.6 struct StdCheatsSafe.Tx1559Detail

```
    struct Tx1559Detail {
        AccessList[] accessList;
        bytes data;
        address from;
        uint256 gas;
        uint256 nonce;
        address to;
        uint256 txType;
        uint256 value;
    }
```

8.8.7 struct StdCheatsSafe.TxLegacy

```
    // Data structures to parse Transaction objects from the broadcast artifact
    // that DO NOT conform to EIP1559. The Raw structs is what is parsed from
      ↪ the JSON
    // and then converted to the one that is used by the user for better UX.

    struct TxLegacy {
        string[] arguments;
        address contractAddress;
        string contractName;
        string functionSig;
        string hash;
        string opcode;
        TxDetailLegacy transaction;
    }
```

8.8.8 struct StdCheatsSafe.TxDetailLegacy

```
    struct TxDetailLegacy {
        AccessList[] accessList;
        uint256 chainId;
        bytes data;
        address from;
        uint256 gas;
        uint256 gasPrice;
        bytes32 hash;
        uint256 nonce;
        bytes1 opcode;
        bytes32 r;
        bytes32 s;
        uint256 txType;
        address to;
        uint8 v;
        uint256 value;
    }
```

8.8.9 struct StdCheatsSafe.AccessList

```
    struct AccessList {
        address accessAddress;
        bytes32[] storageKeys;
    }
```

8.8.10 struct StdCheatsSafe.RawReceipt

```
    // Data structures to parse Receipt objects from the broadcast artifact.
    // The Raw structs is what is parsed from the JSON
    // and then converted to the one that is used by the user for better UX.

    struct RawReceipt {
        bytes32 blockHash;
        bytes blockNumber;
        address contractAddress;
        bytes cumulativeGasUsed;
        bytes effectiveGasPrice;
        address from;
        bytes gasUsed;
        RawReceiptLog[] logs;
        bytes logsBloom;
        bytes status;
        address to;
        bytes32 transactionHash;
        bytes transactionIndex;
    }
```

8.8.11 struct StdCheatsSafe.Receipt

```
    struct Receipt {
        bytes32 blockHash;
        uint256 blockNumber;
        address contractAddress;
        uint256 cumulativeGasUsed;
        uint256 effectiveGasPrice;
        address from;
        uint256 gasUsed;
        ReceiptLog[] logs;
        bytes logsBloom;
        uint256 status;
        address to;
        bytes32 transactionHash;
        uint256 transactionIndex;
    }
```

8.8.12 struct StdCheatsSafe.EIP1559ScriptArtifact

```
    // Data structures to parse the entire broadcast artifact, assuming the
    // transactions conform to EIP1559.

    struct EIP1559ScriptArtifact {
        string[] libraries;
        string path;
        string[] pending;
        Receipt[] receipts;
        uint256 timestamp;
        Tx1559[] transactions;
        TxReturn[] txReturns;
    }
```

8.8.13 struct StdCheatsSafe.RawEIP1559ScriptArtifact

```
    struct RawEIP1559ScriptArtifact {
        string[] libraries;
        string path;
        string[] pending;
        RawReceipt[] receipts;
        TxReturn[] txReturns;
        uint256 timestamp;
        RawTx1559[] transactions;
    }
```

8.8.14 struct StdCheatsSafe.RawReceiptLog

```
    struct RawReceiptLog {
        // json value = address
        address logAddress;
        bytes32 blockHash;
        bytes blockNumber;
        bytes data;
        bytes logIndex;
        bool removed;
        bytes32[] topics;
        bytes32 transactionHash;
        bytes transactionIndex;
        bytes transactionLogIndex;
    }
```

8.8.15 struct StdCheatsSafe.ReceiptLog

```
    struct ReceiptLog {
        // json value = address
        address logAddress;
        bytes32 blockHash;
        uint256 blockNumber;
        bytes data;
        uint256 logIndex;
        bytes32[] topics;
        uint256 transactionIndex;
        uint256 transactionLogIndex;
        bool removed;
    }
```

8.8.16 struct StdCheatsSafe.TxReturn

```
    struct TxReturn {
        string internalType;
        string value;
    }
```

8.8.17 modifier skipWhenForking() [StdCheatsSafe]

```
    modifier skipWhenForking() {
        if (!isFork()) {
            _;
        }
    }
```

8.8.18 modifier skipWhenNotForking() [StdCheatsSafe]

```
    modifier skipWhenNotForking() {
        if (isFork()) {
            _;
        }
    }
```

8.8.19 modifier noGasMetering() [StdCheatsSafe]

```
    modifier noGasMetering() {
        vm.pauseGasMetering();
        // To prevent turning gas monitoring back on with nested functions that
            ↪ use this modifier,
        // we check if gasMetering started in the off position. If it did, we
            ↪ don't want to turn
        // it back on until we exit the top level function that used the
            ↪ modifier
        //
        // i.e. funcA() noGasMetering { funcB() }, where funcB has noGasMetering
            ↪  as well.
        // funcA will have `gasStartedOff` as false, funcB will have it as true,
        // so we only turn metering back on at the end of the funcA
        bool gasStartedOff = gasMeteringOff;
        gasMeteringOff = true;

        _;

        // if gas metering was on when this modifier was called, turn it back on
            ↪  at the end
        if (!gasStartedOff) {
            gasMeteringOff = false;
            vm.resumeGasMetering();
        }
    }
```

8.8.20 _bound(x, min, max) [StdUtils]

```
    function _bound(uint256 x, uint256 min, uint256 max) internal pure virtual
       ↪ returns (uint256 result) {
        require(min <= max, "StdUtils bound(uint256,uint256,uint256): Max is
            ↪ less than min.");
        // If x is between min and max, return x directly. This is to ensure
            ↪ that dictionary values
        // do not get shifted if the min is nonzero. More info: https://github.
            ↪ com/foundry-rs/forge-std/issues/188
        if (x >= min && x <= max) return x;

        uint256 size = max - min + 1;

        // If the value is 0, 1, 2, 3, warp that to min, min+1, min+2, min+3.
            ↪ Similarly for the UINT256_MAX side.
        // This helps ensure coverage of the min/max values.
        if (x <= 3 && size > x) return min + x;
        if (x >= UINT256_MAX - 3 && size > UINT256_MAX - x) return max - (
            ↪ UINT256_MAX - x);

        // Otherwise, wrap x into the range [min, max], i.e. the range is
            ↪ inclusive.
        if (x > max) {
            uint256 diff = x - max;
            uint256 rem = diff % size;
            if (rem == 0) return max;
            result = min + rem - 1;
        } else if (x < min) {
            uint256 diff = min - x;
            uint256 rem = diff % size;
            if (rem == 0) return min;
            result = max - rem + 1;
        }
    }
```

8.8.21 bound(x, min, max) [StdUtils]

```
    function bound(uint256 x, uint256 min, uint256 max) internal view virtual
        ↪ returns (uint256 result) {
        result = _bound(x, min, max);
        console2_log("Bound Result", result);
    }
```

8.8.22 bound(x, min, max) [StdUtils]

```
    function bound(int256 x, int256 min, int256 max) internal view virtual
        ↪ returns (int256 result) {
        require(min <= max, "StdUtils bound(int256,int256,int256): Max is less
            ↪ than min.");

        // Shifting all int256 values to uint256 to use _bound function. The
            ↪ range of two types are:
        // int256 : -(2**255) ~ (2**255 - 1)
        // uint256:     0     ~ (2**256 - 1)
        // So, add 2**255, INT256_MIN_ABS to the integer values.
        //
        // If the given integer value is -2**255, we cannot use `-uint256(-x)`
            ↪ because of the overflow.
        // So, use `~uint256(x) + 1` instead.
        uint256 _x = x < 0 ? (INT256_MIN_ABS - ~uint256(x) - 1) : (uint256(x) +
            ↪ INT256_MIN_ABS);
        uint256 _min = min < 0 ? (INT256_MIN_ABS - ~uint256(min) - 1) : (uint256
            ↪ (min) + INT256_MIN_ABS);
        uint256 _max = max < 0 ? (INT256_MIN_ABS - ~uint256(max) - 1) : (uint256
            ↪ (max) + INT256_MIN_ABS);

        uint256 y = _bound(_x, _min, _max);

        // To move it back to int256 value, subtract INT256_MIN_ABS at here.
        result = y < INT256_MIN_ABS ? int256(~(INT256_MIN_ABS - y) + 1) : int256
            ↪ (y - INT256_MIN_ABS);
        console2_log("Bound result", vm.toString(result));
    }
```

8.8.23 computeCreateAddress(deployer, nonce) [StdUtils]

```
    /// @dev Compute the address a contract will be deployed at for a given
        ↪ deployer address and nonce
    /// @notice adapated from Solmate implementation (https://github.com/Rari-
        ↪ Capital/solmate/blob/main/src/utils/LibRLP.sol)
    function computeCreateAddress(address deployer, uint256 nonce) internal pure
        ↪  virtual returns (address) {
        // forgefmt: disable-start
        // The integer zero is treated as an empty byte string, and as a result
            ↪ it only has a length prefix, 0x80, computed via 0x80 + 0.
        // A one byte integer uses its own value as its length prefix, there is
            ↪ no additional "0x80 + length" prefix that comes before it.
        if (nonce == 0x00)      return addressFromLast20Bytes(keccak256(abi.
            ↪ encodePacked(bytes1(0xd6), bytes1(0x94), deployer, bytes1(0x80)))
            ↪ );
        if (nonce <= 0x7f)      return addressFromLast20Bytes(keccak256(abi.
            ↪ encodePacked(bytes1(0xd6), bytes1(0x94), deployer, uint8(nonce)))
            ↪ );
        // Nonces greater than 1 byte all follow a consistent encoding scheme,
            ↪ where each value is preceded by a prefix of 0x80 + length.
        if (nonce <= 2**8 - 1)  return addressFromLast20Bytes(keccak256(abi.
            ↪ encodePacked(bytes1(0xd7), bytes1(0x94), deployer, bytes1(0x81),
            ↪ uint8(nonce))));
        if (nonce <= 2**16 - 1) return addressFromLast20Bytes(keccak256(abi.
            ↪ encodePacked(bytes1(0xd8), bytes1(0x94), deployer, bytes1(0x82),
            ↪ uint16(nonce))));
        if (nonce <= 2**24 - 1) return addressFromLast20Bytes(keccak256(abi.
            ↪ encodePacked(bytes1(0xd9), bytes1(0x94), deployer, bytes1(0x83),
            ↪ uint24(nonce))));
        // forgefmt: disable-end

        // More details about RLP encoding can be found here: https://eth.wiki/
            ↪ fundamentals/rlp
        // 0xda = 0xc0 (short RLP prefix) + 0x16 (length of: 0x94 ++ proxy ++ 0
            ↪ x84 ++ nonce)
        // 0x94 = 0x80 + 0x14 (0x14 = the length of an address, 20 bytes, in hex
            ↪ )
        // 0x84 = 0x80 + 0x04 (0x04 = the bytes length of the nonce, 4 bytes, in
            ↪  hex)
        // We assume nobody can have a nonce large enough to require more than
            ↪ 32 bytes.
        return addressFromLast20Bytes(
            keccak256(abi.encodePacked(bytes1(0xda), bytes1(0x94), deployer,
                ↪ bytes1(0x84), uint32(nonce)))
        );
    }
```

8.8.24 computeCreate2Address(salt, initcodeHash, deployer) [StdUtils]

```
    function computeCreate2Address(bytes32 salt, bytes32 initcodeHash, address
        ↪ deployer)
        internal
        pure
        virtual
        returns (address)
    {
        return addressFromLast20Bytes(keccak256(abi.encodePacked(bytes1(0xff),
            ↪ deployer, salt, initcodeHash)));
    }
```

8.8.25 bytesToUint(b) [StdUtils]

```
    function bytesToUint(bytes memory b) internal pure virtual returns (uint256)
        ↪  {
        require(b.length <= 32, "StdUtils bytesToUint(bytes): Bytes length
            ↪ exceeds 32.");
        return abi.decode(abi.encodePacked(new bytes(32 - b.length), b), (
            ↪ uint256));
    }
```

8.8.26 addressFromLast20Bytes(bytesValue) [StdUtils]

```
    function addressFromLast20Bytes(bytes32 bytesValue) private pure returns (
        ↪ address) {
        return address(uint160(uint256(bytesValue)));
    }
```

8.8.27 console2_log(p0, p1) [StdUtils]

```
    // Used to prevent the compilation of console, which shortens the
       ↪ compilation time when console is not used elsewhere.

    function console2_log(string memory p0, uint256 p1) private view {
        (bool status,) = address(CONSOLE2_ADDRESS).staticcall(abi.
            ↪ encodeWithSignature("log(string,uint256)", p0, p1));
        status;
    }
```

8.8.28 console2_log(p0, p1) [StdUtils]

```
    function console2_log(string memory p0, string memory p1) private view {
        (bool status,) = address(CONSOLE2_ADDRESS).staticcall(abi.
            ↪ encodeWithSignature("log(string,string)", p0, p1));
        status;
    }
```

8.8.29 assumeNoPrecompiles(addr) [StdCheatsSafe]

```
    function assumeNoPrecompiles(address addr) internal virtual {
        // Assembly required since `block.chainid` was introduced in 0.8.0.
        uint256 chainId;
        assembly {
            chainId := chainid()
        }
        assumeNoPrecompiles(addr, chainId);
    }
```

8.8.30 assumeNoPrecompiles(addr, chainId) [StdCheatsSafe]

```
    function assumeNoPrecompiles(address addr, uint256 chainId) internal pure
       ↪ virtual {
        // Note: For some chains like Optimism these are technically predeploys
            ↪ (i.e. bytecode placed at a specific
        // address), but the same rationale for excluding them applies so we
            ↪ include those too.

        // These should be present on all EVM-compatible chains.
        vm.assume(addr < address(0x1) || addr > address(0x9));

        // forgefmt: disable-start
        if (chainId == 10 || chainId == 420) {
            // https://github.com/ethereum-optimism/optimism/blob/
                ↪ eaa371a0184b56b7ca6d9eb9cb0a2b78b2ccd864/op-bindings/
                ↪ predeploys/addresses.go#L6-L21
            vm.assume(addr < address(0x4200000000000000000000000000000000000000)
                ↪  || addr > address(0x4200000000000000000000000000000000000800
                ↪ ));
        } else if (chainId == 42161 || chainId == 421613) {
            // https://developer.arbitrum.io/useful-addresses#arbitrum-
                ↪ precompiles-l2-same-on-all-arb-chains
            vm.assume(addr < address(0x0000000000000000000000000000000000000064)
                ↪  || addr > address(0x0000000000000000000000000000000000000068
                ↪ ));
        } else if (chainId == 43114 || chainId == 43113) {
            // https://github.com/ava-labs/subnet-evm/blob/47
                ↪ c03fd007ecaa6de2c52ea081596e0a88401f58/precompile/params.go#
                ↪ L18-L59
            vm.assume(addr < address(0x0100000000000000000000000000000000000000)
                ↪  || addr > address(0x01000000000000000000000000000000000000ff
                ↪ ));
            vm.assume(addr < address(0x0200000000000000000000000000000000000000)
                ↪  || addr > address(0x02000000000000000000000000000000000000FF
                ↪ ));
            vm.assume(addr < address(0x0300000000000000000000000000000000000000)
                ↪  || addr > address(0x03000000000000000000000000000000000000Ff
                ↪ ));
        }
        // forgefmt: disable-end
    }
```

8.8.31 readEIP1559ScriptArtifact(path) [StdCheatsSafe]

```
    function readEIP1559ScriptArtifact(string memory path)
        internal
        view
        virtual
        returns (EIP1559ScriptArtifact memory)
    {
        string memory data = vm.readFile(path);
        bytes memory parsedData = vm.parseJson(data);
        RawEIP1559ScriptArtifact memory rawArtifact = abi.decode(parsedData, (
            ↪ RawEIP1559ScriptArtifact));
        EIP1559ScriptArtifact memory artifact;
        artifact.libraries = rawArtifact.libraries;
        artifact.path = rawArtifact.path;
        artifact.timestamp = rawArtifact.timestamp;
        artifact.pending = rawArtifact.pending;
        artifact.txReturns = rawArtifact.txReturns;
        artifact.receipts = rawToConvertedReceipts(rawArtifact.receipts);
        artifact.transactions = rawToConvertedEIPTx1559s(rawArtifact.
            ↪ transactions);
        return artifact;
    }
```

8.8.32 rawToConvertedEIPTx1559s(rawTxs) [StdCheatsSafe]

```
    function rawToConvertedEIPTx1559s(RawTx1559[] memory rawTxs) internal pure
        ↪ virtual returns (Tx1559[] memory) {
        Tx1559[] memory txs = new Tx1559[](rawTxs.length);
        for (uint256 i; i < rawTxs.length; i++) {
            txs[i] = rawToConvertedEIPTx1559(rawTxs[i]);
        }
        return txs;
    }
```

8.8.33 rawToConvertedEIPTx1559(rawTx) [StdCheatsSafe]

```
    function rawToConvertedEIPTx1559(RawTx1559 memory rawTx) internal pure
        ↪ virtual returns (Tx1559 memory) {
        Tx1559 memory transaction;
        transaction.arguments = rawTx.arguments;
        transaction.contractName = rawTx.contractName;
        transaction.functionSig = rawTx.functionSig;
        transaction.hash = rawTx.hash;
        transaction.txDetail = rawToConvertedEIP1559Detail(rawTx.txDetail);
        transaction.opcode = rawTx.opcode;
        return transaction;
    }
```

8.8.34 rawToConvertedEIP1559Detail(rawDetail) [StdCheatsSafe]

```
    function rawToConvertedEIP1559Detail(RawTx1559Detail memory rawDetail)
        internal
        pure
        virtual
        returns (Tx1559Detail memory)
    {
        Tx1559Detail memory txDetail;
        txDetail.data = rawDetail.data;
        txDetail.from = rawDetail.from;
        txDetail.to = rawDetail.to;
        txDetail.nonce = _bytesToUint(rawDetail.nonce);
        txDetail.txType = _bytesToUint(rawDetail.txType);
        txDetail.value = _bytesToUint(rawDetail.value);
        txDetail.gas = _bytesToUint(rawDetail.gas);
        txDetail.accessList = rawDetail.accessList;
        return txDetail;
    }
```

8.8.35 readTx1559s(path) [StdCheatsSafe]

```
    function readTx1559s(string memory path) internal view virtual returns (
        ↪ Tx1559[] memory) {
        string memory deployData = vm.readFile(path);
        bytes memory parsedDeployData = vm.parseJson(deployData, ".transactions"
            ↪ );
        RawTx1559[] memory rawTxs = abi.decode(parsedDeployData, (RawTx1559[]));
        return rawToConvertedEIPTx1559s(rawTxs);
    }
```

8.8.36 readTx1559(path, index) [StdCheatsSafe]

```
    function readTx1559(string memory path, uint256 index) internal view virtual
        ↪  returns (Tx1559 memory) {
        string memory deployData = vm.readFile(path);
        string memory key = string(abi.encodePacked(".transactions[", vm.
            ↪ toString(index), "]"));
        bytes memory parsedDeployData = vm.parseJson(deployData, key);
        RawTx1559 memory rawTx = abi.decode(parsedDeployData, (RawTx1559));
        return rawToConvertedEIPTx1559(rawTx);
    }
```

8.8.37 readReceipts(path) [StdCheatsSafe]

```
    // Analogous to readTransactions, but for receipts.
    function readReceipts(string memory path) internal view virtual returns (
        ↪ Receipt[] memory) {
        string memory deployData = vm.readFile(path);
        bytes memory parsedDeployData = vm.parseJson(deployData, ".receipts");
        RawReceipt[] memory rawReceipts = abi.decode(parsedDeployData, (
            ↪ RawReceipt[]));
        return rawToConvertedReceipts(rawReceipts);
    }
```

8.8.38 readReceipt(path, index) [StdCheatsSafe]

```
    function readReceipt(string memory path, uint256 index) internal view
        ↪ virtual returns (Receipt memory) {
        string memory deployData = vm.readFile(path);
        string memory key = string(abi.encodePacked(".receipts[", vm.toString(
            ↪ index), "]"));
        bytes memory parsedDeployData = vm.parseJson(deployData, key);
        RawReceipt memory rawReceipt = abi.decode(parsedDeployData, (RawReceipt)
            ↪ );
        return rawToConvertedReceipt(rawReceipt);
    }
```

8.8.39 rawToConvertedReceipts(rawReceipts) [StdCheatsSafe]

```
function rawToConvertedReceipts(RawReceipt[] memory rawReceipts) internal
    ↪ pure virtual returns (Receipt[] memory) {
    Receipt[] memory receipts = new Receipt[](rawReceipts.length);
    for (uint256 i; i < rawReceipts.length; i++) {
        receipts[i] = rawToConvertedReceipt(rawReceipts[i]);
    }
    return receipts;
}
```

8.8.40 rawToConvertedReceipt(rawReceipt) [StdCheatsSafe]

```
function rawToConvertedReceipt(RawReceipt memory rawReceipt) internal pure
    ↪ virtual returns (Receipt memory) {
    Receipt memory receipt;
    receipt.blockHash = rawReceipt.blockHash;
    receipt.to = rawReceipt.to;
    receipt.from = rawReceipt.from;
    receipt.contractAddress = rawReceipt.contractAddress;
    receipt.effectiveGasPrice = _bytesToUint(rawReceipt.effectiveGasPrice);
    receipt.cumulativeGasUsed = _bytesToUint(rawReceipt.cumulativeGasUsed);
    receipt.gasUsed = _bytesToUint(rawReceipt.gasUsed);
    receipt.status = _bytesToUint(rawReceipt.status);
    receipt.transactionIndex = _bytesToUint(rawReceipt.transactionIndex);
    receipt.blockNumber = _bytesToUint(rawReceipt.blockNumber);
    receipt.logs = rawToConvertedReceiptLogs(rawReceipt.logs);
    receipt.logsBloom = rawReceipt.logsBloom;
    receipt.transactionHash = rawReceipt.transactionHash;
    return receipt;
}
```

8.8.41 rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe]

```
function rawToConvertedReceiptLogs(RawReceiptLog[] memory rawLogs)
    internal
    pure
    virtual
    returns (ReceiptLog[] memory)
{
    ReceiptLog[] memory logs = new ReceiptLog[](rawLogs.length);
    for (uint256 i; i < rawLogs.length; i++) {
        logs[i].logAddress = rawLogs[i].logAddress;
        logs[i].blockHash = rawLogs[i].blockHash;
        logs[i].blockNumber = _bytesToUint(rawLogs[i].blockNumber);
        logs[i].data = rawLogs[i].data;
        logs[i].logIndex = _bytesToUint(rawLogs[i].logIndex);
        logs[i].topics = rawLogs[i].topics;
        logs[i].transactionIndex = _bytesToUint(rawLogs[i].transactionIndex)
            ↪ ;
        logs[i].transactionLogIndex = _bytesToUint(rawLogs[i].
            ↪ transactionLogIndex);
        logs[i].removed = rawLogs[i].removed;
    }
    return logs;
}
```

8.8.42 deployCode(what, args) [StdCheatsSafe]

```
// Deploy a contract by fetching the contract bytecode from
// the artifacts directory
// e.g. `deployCode(code, abi.encode(arg1,arg2,arg3))`
function deployCode(string memory what, bytes memory args) internal virtual
    ↪ returns (address addr) {
        bytes memory bytecode = abi.encodePacked(vm.getCode(what), args);
        /// @solidity memory-safe-assembly
        assembly {
            addr := create(0, add(bytecode, 0x20), mload(bytecode))
        }

        require(addr != address(0), "StdCheats deployCode(string,bytes):
            ↪ Deployment failed.");
    }
```

8.8.43 deployCode(what) [StdCheatsSafe]

```
    function deployCode(string memory what) internal virtual returns (address
        ↪ addr) {
        bytes memory bytecode = vm.getCode(what);
        /// @solidity memory-safe-assembly
        assembly {
            addr := create(0, add(bytecode, 0x20), mload(bytecode))
        }

        require(addr != address(0), "StdCheats deployCode(string): Deployment
            ↪ failed.");
    }
```

8.8.44 deployCode(what, args, val) [StdCheatsSafe]

```
    /// @dev deploy contract with value on construction
    function deployCode(string memory what, bytes memory args, uint256 val)
        ↪ internal virtual returns (address addr) {
        bytes memory bytecode = abi.encodePacked(vm.getCode(what), args);
        /// @solidity memory-safe-assembly
        assembly {
            addr := create(val, add(bytecode, 0x20), mload(bytecode))
        }

        require(addr != address(0), "StdCheats deployCode(string,bytes,uint256):
            ↪  Deployment failed.");
    }
```

8.8.45 deployCode(what, val) [StdCheatsSafe]

```
    function deployCode(string memory what, uint256 val) internal virtual
        ↪ returns (address addr) {
        bytes memory bytecode = vm.getCode(what);
        /// @solidity memory-safe-assembly
        assembly {
            addr := create(val, add(bytecode, 0x20), mload(bytecode))
        }

        require(addr != address(0), "StdCheats deployCode(string,uint256):
            ↪ Deployment failed.");
    }
```

8.8.46 makeAddrAndKey(name) [StdCheatsSafe]

```
    // creates a labeled address and the corresponding private key
    function makeAddrAndKey(string memory name) internal virtual returns (
        ↪ address addr, uint256 privateKey) {
        privateKey = uint256(keccak256(abi.encodePacked(name)));
        addr = vm.addr(privateKey);
        vm.label(addr, name);
    }
```

8.8.47 makeAddr(name) [StdCheatsSafe]

```
// creates a labeled address
function makeAddr(string memory name) internal virtual returns (address addr
    ↪ ) {
    (addr,) = makeAddrAndKey(name);
}
```

8.8.48 deriveRememberKey(mnemonic, index) [StdCheatsSafe]

```
function deriveRememberKey(string memory mnemonic, uint32 index)
    internal
    virtual
    returns (address who, uint256 privateKey)
{
    privateKey = vm.deriveKey(mnemonic, index);
    who = vm.rememberKey(privateKey);
}
```

8.8.49 _bytesToUint(b) [StdCheatsSafe]

```
function _bytesToUint(bytes memory b) private pure returns (uint256) {
    require(b.length <= 32, "StdCheats _bytesToUint(bytes): Bytes length
        ↪ exceeds 32.");
    return abi.decode(abi.encodePacked(new bytes(32 - b.length), b), (
        ↪ uint256));
}
```

8.8.50 isFork() [StdCheatsSafe]

```
function isFork() internal view virtual returns (bool status) {
    try vm.activeFork() {
        status = true;
    } catch (bytes memory) {}
}
```

8.8.51 getChain(chainAlias) [StdChains]

```
// The RPC URL will be fetched from config or defaultRpcUrls if possible.
function getChain(string memory chainAlias) internal virtual returns (Chain
    ↪ memory chain) {
    require(bytes(chainAlias).length != 0, "StdChains getChain(string):
        ↪ Chain alias cannot be the empty string.");

    initialize();
    chain = chains[chainAlias];
    require(
        chain.chainId != 0,
        string(abi.encodePacked("StdChains getChain(string): Chain with
            ↪ alias \"", chainAlias, "\" not found."))
    );

    chain = getChainWithUpdatedRpcUrl(chainAlias, chain);
}
```

8.8.52 getChain(chainId) [StdChains]

```
function getChain(uint256 chainId) internal virtual returns (Chain memory
    ↪ chain) {
    require(chainId != 0, "StdChains getChain(uint256): Chain ID cannot be
        ↪ 0.");
        initialize();
        string memory chainAlias = idToAlias[chainId];

        chain = chains[chainAlias];

        require(
            chain.chainId != 0,
            string(abi.encodePacked("StdChains getChain(uint256): Chain with ID
                ↪ ", vm.toString(chainId), " not found."))
        );

        chain = getChainWithUpdatedRpcUrl(chainAlias, chain);
    }
```

8.8.53 setChain(chainAlias, chain) [StdChains]

```
    // set chain info, with priority to argument's rpcUrl field.
    function setChain(string memory chainAlias, ChainData memory chain) internal
        ↪ virtual {
        require(
            bytes(chainAlias).length != 0,
            "StdChains setChain(string,ChainData): Chain alias cannot be the
                ↪ empty string."
        );

        require(chain.chainId != 0, "StdChains setChain(string,ChainData): Chain
            ↪ ID cannot be 0.");

        initialize();
        string memory foundAlias = idToAlias[chain.chainId];

        require(
            bytes(foundAlias).length == 0 || keccak256(bytes(foundAlias)) ==
                ↪ keccak256(bytes(chainAlias)),
            string(
                abi.encodePacked(
                    "StdChains setChain(string,ChainData): Chain ID ",
                    vm.toString(chain.chainId),
                    " already used by \"",
                    foundAlias,
                    "\"."
                )
            )
        );

        uint256 oldChainId = chains[chainAlias].chainId;
        delete idToAlias[oldChainId];

        chains[chainAlias] =
            Chain({name: chain.name, chainId: chain.chainId, chainAlias:
                ↪ chainAlias, rpcUrl: chain.rpcUrl});
        idToAlias[chain.chainId] = chainAlias;
    }
```

8.8.54 setChain(chainAlias, chain) [StdChains]

```
    // set chain info, with priority to argument's rpcUrl field.
    function setChain(string memory chainAlias, Chain memory chain) internal
        ↪ virtual {
        setChain(chainAlias, ChainData({name: chain.name, chainId: chain.chainId
            ↪ , rpcUrl: chain.rpcUrl}));
    }
```

8.8.55 _toUpper(str) [StdChains]

```
function _toUpper(string memory str) private pure returns (string memory) {
    bytes memory strb = bytes(str);
    bytes memory copy = new bytes(strb.length);
    for (uint256 i = 0; i < strb.length; i++) {
        bytes1 b = strb[i];
        if (b >= 0x61 && b <= 0x7A) {
            copy[i] = bytes1(uint8(b) - 32);
        } else {
            copy[i] = b;
        }
    }
    return string(copy);
}
```

8.8.56 getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains]

```
// lookup rpcUrl, in descending order of priority:
// current -> config (foundry.toml) -> environment variable -> default
function getChainWithUpdatedRpcUrl(string memory chainAlias, Chain memory
   ↪ chain) private returns (Chain memory) {
    if (bytes(chain.rpcUrl).length == 0) {
        try vm.rpcUrl(chainAlias) returns (string memory configRpcUrl) {
            chain.rpcUrl = configRpcUrl;
        } catch (bytes memory err) {
            chain.rpcUrl =
                vm.envOr(string(abi.encodePacked(_toUpper(chainAlias), "
                   ↪ _RPC_URL")), defaultRpcUrls[chainAlias]);
            // distinguish 'not found' from 'cannot read'
            bytes memory notFoundError =
                abi.encodeWithSignature("CheatCodeError", string(abi.
                   ↪ encodePacked("invalid rpc url ", chainAlias)));
            if (keccak256(notFoundError) != keccak256(err) || bytes(chain.
               ↪ rpcUrl).length == 0) {
                /// @solidity memory-safe-assembly
                assembly {
                    revert(add(32, err), mload(err))
                }
            }
        }
    }
    return chain;
}
```

8.8.57 initialize() [StdChains]

```
function initialize() private {
    if (initialized) return;

    initialized = true;

    // If adding an RPC here, make sure to test the default RPC URL in `
       ↪ testRpcs`
    setChainWithDefaultRpcUrl("anvil", ChainData("Anvil", 31337, "http
       ↪ ://127.0.0.1:8545"));
    setChainWithDefaultRpcUrl(
        "mainnet", ChainData("Mainnet", 1, "https://mainnet.infura.io/v3
           ↪ /6770454bc6ea42c58aac12978531b93f")
    );
    setChainWithDefaultRpcUrl(
        "goerli", ChainData("Goerli", 5, "https://goerli.infura.io/v3
           ↪ /6770454bc6ea42c58aac12978531b93f")
    );
    setChainWithDefaultRpcUrl(
        "sepolia", ChainData("Sepolia", 11155111, "https://sepolia.infura.io
           ↪ /v3/6770454bc6ea42c58aac12978531b93f")
    );
    setChainWithDefaultRpcUrl("optimism", ChainData("Optimism", 10, "https
       ↪ ://mainnet.optimism.io"));
    setChainWithDefaultRpcUrl("optimism_goerli", ChainData("Optimism Goerli"
       ↪ , 420, "https://goerli.optimism.io"));
    setChainWithDefaultRpcUrl("arbitrum_one", ChainData("Arbitrum One",
       ↪ 42161, "https://arb1.arbitrum.io/rpc"));
    setChainWithDefaultRpcUrl(
        "arbitrum_one_goerli", ChainData("Arbitrum One Goerli", 421613, "
           ↪ https://goerli-rollup.arbitrum.io/rpc")
    );
    setChainWithDefaultRpcUrl("arbitrum_nova", ChainData("Arbitrum Nova",
       ↪ 42170, "https://nova.arbitrum.io/rpc"));
    setChainWithDefaultRpcUrl("polygon", ChainData("Polygon", 137, "https://
       ↪ polygon-rpc.com"));
    setChainWithDefaultRpcUrl(
        "polygon_mumbai", ChainData("Polygon Mumbai", 80001, "https://rpc-
           ↪ mumbai.maticvigil.com")
    );
    setChainWithDefaultRpcUrl("avalanche", ChainData("Avalanche", 43114, "
       ↪ https://api.avax.network/ext/bc/C/rpc"));
    setChainWithDefaultRpcUrl(
        "avalanche_fuji", ChainData("Avalanche Fuji", 43113, "https://api.
           ↪ avax-test.network/ext/bc/C/rpc")
    );
    setChainWithDefaultRpcUrl(
        "bnb_smart_chain", ChainData("BNB Smart Chain", 56, "https://bsc-
           ↪ dataseed1.binance.org")
    );
    setChainWithDefaultRpcUrl(
        "bnb_smart_chain_testnet",
        ChainData("BNB Smart Chain Testnet", 97, "https://data-seed-prebsc
           ↪ -1-s1.binance.org:8545")
    );
    setChainWithDefaultRpcUrl("gnosis_chain", ChainData("Gnosis Chain", 100,
       ↪  "https://rpc.gnosischain.com"));
}
```

8.8.58 setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains]

```
    // set chain info, with priority to chainAlias' rpc url in foundry.toml
    function setChainWithDefaultRpcUrl(string memory chainAlias, ChainData
        ↪ memory chain) private {
        string memory rpcUrl = chain.rpcUrl;
        defaultRpcUrls[chainAlias] = rpcUrl;
        chain.rpcUrl = "";
        setChain(chainAlias, chain);
        chain.rpcUrl = rpcUrl; // restore argument
    }
```

8.8.59 run() X

```
    function run() external {
        config = ScriptTools.loadConfig(NAME);
        dependencies = ScriptTools.loadDependencies(NAME);
        dss = MCD.loadFromChainlog(config.readAddress("chainlog"));

        d3mCore = D3MCoreInstance({
            hub: dependencies.readAddress("hub"),
            mom: dependencies.readAddress("mom")
        });

        vm.startBroadcast();
        D3MInit.initCore(
            dss,
            d3mCore
        );
        vm.stopBroadcast();
    }
```

8.9 contract D3MCompoundPool

```
contract D3MCompoundPool is ID3MPool {

    mapping (address => uint256) public wards;
    address                      public hub;
    address                      public king; // Who gets the rewards
    uint256                      public exited;

    bytes32            public immutable ilk;
    VatLike            public immutable vat;
    ComptrollerLike public immutable comptroller;
    TokenLike          public immutable comp;
    TokenLike          public immutable dai;
    CErc20Like         public immutable cDai;

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event File(bytes32 indexed what, address data);
    event Collect(address indexed king, address indexed gift, uint256 amt);

    // --- Math ---
    uint256 internal constant WAD = 10 ** 18;

    function postDebtChange() external override {}
}
```

8.9.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "D3MCompoundPool/not-authorized");
        _;
    }
```

8.9.2 modifier onlyHub()

```
    modifier onlyHub {
        require(msg.sender == hub, "D3MCompoundPool/only-hub");
        _;
    }
```

8.9.3 constructor(ilk_, hub_, cDai_) X

```
    constructor(bytes32 ilk_, address hub_, address cDai_) {
        ilk           = ilk_;
        cDai          = CErc20Like(cDai_);
        dai           = TokenLike(cDai.underlying());
        comptroller = ComptrollerLike(cDai.comptroller());
        comp          = TokenLike(comptroller.getCompAddress());

        require(address(comp) != address(0), "D3MCompoundPool/invalid-comp");

        dai.approve(cDai_, type(uint256).max);

        hub = hub_;
        vat = VatLike(D3mHubLike(hub_).vat());
        vat.hope(hub_);

        wards[msg.sender] = 1;
        emit Rely(msg.sender);
    }
```

8.9.4 _wmul(x, y)

```
function _wmul(uint256 x, uint256 y) internal pure returns (uint256 z) {
    z = (x * y) / WAD;
}
```

8.9.5 _wdiv(x, y)

```
function _wdiv(uint256 x, uint256 y) internal pure returns (uint256 z) {
    z = (x * WAD) / y;
}
```

8.9.6 _min(x, y)

```
function _min(uint256 x, uint256 y) internal pure returns (uint256 z) {
    z = x <= y ? x : y;
}
```

8.9.7 rely(usr) X a

```
// --- Admin ---
function rely(address usr) external auth {
    wards[usr] = 1;
    emit Rely(usr);
}
```

8.9.8 deny(usr) X a

```
function deny(address usr) external auth {
    wards[usr] = 0;
    emit Deny(usr);
}
```

8.9.9 file(what, data) X a

```
function file(bytes32 what, address data) external auth {
    require(vat.live() == 1, "D3MCompoundPool/no-file-during-shutdown");
    if (what == "hub") {
        vat.nope(hub);
        hub = data;
        vat.hope(data);
    } else if (what == "king") king = data;
    else revert("D3MCompoundPool/file-unrecognized-param");
    emit File(what, data);
}
```

8.9.10 deposit(wad) X

```
function deposit(uint256 wad) external override onlyHub {
    uint256 prev = cDai.balanceOf(address(this));
    require(cDai.mint(wad) == 0, "D3MCompoundPool/mint-failure");

    // As interest was accrued on `mint` we can use the non accruing `
        ↪ exchangeRateStored`
    require(
        cDai.balanceOf(address(this)) ==
        prev + _wdiv(wad, cDai.exchangeRateStored()), "D3MCompoundPool/
            ↪ incorrect-cdai-credit"
    );
}
```

8.9.11 withdraw(wad) X

```
function withdraw(uint256 wad) external override onlyHub {
    uint256 prevDai = dai.balanceOf(msg.sender);

    require(cDai.redeemUnderlying(wad) == 0, "D3MCompoundPool/
        ↪ redeemUnderlying-failure");
    dai.transfer(msg.sender, wad);

    require(dai.balanceOf(msg.sender) == prevDai + wad, "D3MCompoundPool/
        ↪ incorrect-dai-balance-received");
}
```

8.9.12 exit(dst, wad) X

```
function exit(address dst, uint256 wad) external override onlyHub {
    uint256 exited_ = exited;
    exited = exited_ + wad;
    uint256 amt = wad * cDai.balanceOf(address(this)) / (D3mHubLike(hub).end
        ↪ ().Art(ilk) - exited_);
    require(cDai.transfer(dst, amt), "D3MCompoundPool/transfer-failed");
}
```

8.9.13 quit(dst) X a

```
function quit(address dst) external override auth {
    require(vat.live() == 1, "D3MCompoundPool/no-quit-during-shutdown");
    require(cDai.transfer(dst, cDai.balanceOf(address(this))), "
        ↪ D3MCompoundPool/transfer-failed");
}
```

8.9.14 preDebtChange() X

```
function preDebtChange() external override {
    require(cDai.accrueInterest() == 0, "D3MCompoundPool/accrueInterest-
        ↪ failure");
}
```

8.9.15 assetBalance()

```
// Does not accrue interest (as opposed to cToken's balanceOfUnderlying()
    ↪ which is not a view function).
function assetBalance() public view override returns (uint256) {
    (uint256 error, uint256 cTokenBalance,, uint256 exchangeRate) = cDai.
        ↪ getAccountSnapshot(address(this));
    require(error == 0, "D3MCompoundPool/getAccountSnapshot-failure");
    return _wmul(cTokenBalance, exchangeRate);
}
```

8.9.16 maxDeposit()

```
function maxDeposit() external pure override returns (uint256) {
    return type(uint256).max;
}
```

8.9.17 maxWithdraw()

```
function maxWithdraw() external view override returns (uint256) {
    return _min(cDai.getCash(), assetBalance());
}
```

8.9.18 redeemable()

```
    function redeemable() external view override returns (address) {
        return address(cDai);
    }
```

8.9.19 collect(claim) X

```
    function collect(bool claim) external {
        require(king != address(0), "D3MCompoundPool/king-not-set");

        if (claim) {
            address[] memory holders = new address[](1);
            holders[0] = address(this);
            address[] memory cTokens = new address[](1);
            cTokens[0] = address(cDai);
            comptroller.claimComp(holders, cTokens, false, true);
        }

        uint256 amt = comp.balanceOf(address(this));
        comp.transfer(king, amt);

        emit Collect(king, address(comp), amt);
    }
```

8.10 contract OptionalLoadDependencies

```
contract OptionalLoadDependencies {

}
```

8.10.1 loadDependencies(name) X

```
    function loadDependencies(string memory name) external returns (string
        ↪ memory) {
        return ScriptTools.loadDependencies(name);
    }
```

8.11 contract D3MDeployScript

```
contract D3MDeployScript is Script {

    using stdJson for string;
    using ScriptTools for string;

    string config;
    string dependencies;
    DssInstance dss;

    string d3mType;
    address admin;
    address hub;
    bytes32 ilk;
    D3MInstance d3m;

}
```

Inherited:

```
// ?? SCRIPT
abstract contract Script is StdChains, StdCheatsSafe, StdUtils, ScriptBase {
    // Note: IS_SCRIPT() must return true.
    bool public IS_SCRIPT = true;
}
```

```
abstract contract ScriptBase is CommonBase {
    // Used when deploying with create2, https://github.com/Arachnid/
        ↪ deterministic-deployment-proxy.
    address internal constant CREATE2_FACTORY = 0
        ↪ x4e59b44847b379578588920cA78FbF26c0B4956C;

    VmSafe internal constant vmSafe = VmSafe(VM_ADDRESS);
}
```

```
abstract contract CommonBase {
    // Cheat code address, 0x7109709ECfa91a80626fF3989D68f67F5b1DD12D.
    address internal constant VM_ADDRESS = address(uint160(uint256(keccak256("
        ↪ hevm cheat code"))));
    // console.sol and console2.sol work by executing a staticcall to this
        ↪ address.
    address internal constant CONSOLE = 0
        ↪ x000000000000000000636F6e736F6c652e6c6f67;
    // Default address for tx.origin and msg.sender, 0
        ↪ x1804c8AB1F12E6bbf3894d4083f33e07309d1f38.
    address internal constant DEFAULT_SENDER = address(uint160(uint256(keccak256
        ↪ ("foundry default caller"))));
    // Address of the test contract, deployed by the DEFAULT_SENDER.
    address internal constant DEFAULT_TEST_CONTRACT = 0
        ↪ x5615dEB798BB3E4dFa0139dFa1b3D433Cc23b72f;
    // Deterministic deployment address of the Multicall3 contract.
    address internal constant MULTICALL3_ADDRESS = 0
        ↪ xcA11bde05977b3631167028862bE2a173976CA11;

    uint256 internal constant UINT256_MAX =
        1157920892373161954235709850086879078532699846656405640394575840079131296399935
            ↪

    Vm internal constant vm = Vm(VM_ADDRESS);
    StdStorage internal stdstore;
}
```

```
abstract contract StdUtils {
    VmSafe private constant vm = VmSafe(address(uint160(uint256(keccak256("hevm
        ↪ cheat code")))));
    address private constant CONSOLE2_ADDRESS = 0
        ↪ x000000000000000000636F6e736F6c652e6c6f67;
    uint256 private constant INT256_MIN_ABS =
        57896044618658097711785492504343953926634992332820282019728792003956564819968;
           ↪
    uint256 private constant UINT256_MAX =
        115792089237316195423570985008687907853269984665640564039457584007913129639935;
           ↪
}
```

```
abstract contract StdCheatsSafe {
    Vm private constant vm = Vm(address(uint160(uint256(keccak256("hevm cheat
       ↪ code")))));

    bool private gasMeteringOff;
}
```

```
/**
 * StdChains provides information about EVM compatible chains that can be used
    ↪ in scripts/tests.
 * For each chain, the chain's name, chain ID, and a default RPC URL are
    ↪ provided. Chains are
 * identified by their alias, which is the same as the alias in the `[
    ↪ rpc_endpoints]` section of
 * the `foundry.toml` file. For best UX, ensure the alias in the `foundry.toml`
    ↪ file match the
 * alias used in this contract, which can be found as the first argument to the
 * `setChainWithDefaultRpcUrl` call in the `initialize` function.
 *
 * There are two main ways to use this contract:
 *   1. Set a chain with `setChain(string memory chainAlias, ChainData memory
    ↪ chain)` or
 *      `setChain(string memory chainAlias, Chain memory chain)`
 *   2. Get a chain with `getChain(string memory chainAlias)` or `getChain(
    ↪ uint256 chainId)`.
 *
 * The first time either of those are used, chains are initialized with the
    ↪ default set of RPC URLs.
 * This is done in `initialize`, which uses `setChainWithDefaultRpcUrl`.
    ↪ Defaults are recorded in
 * `defaultRpcUrls`.
 *
 * The `setChain` function is straightforward, and it simply saves off the given
    ↪  chain data.
 *
 * The `getChain` methods use `getChainWithUpdatedRpcUrl` to return a chain. For
    ↪  example, let's say
 * we want to retrieve `mainnet`'s RPC URL:
 *   - If you haven't set any mainnet chain info with `setChain`, you haven't
    ↪ specified that
 *     chain in `foundry.toml` and no env var is set, the default data and RPC
    ↪ URL will be returned.
 *   - If you have set a mainnet RPC URL in `foundry.toml` it will return that,
    ↪ if valid (e.g. if
 *     a URL is given or if an environment variable is given and that
    ↪ environment variable exists).
 *     Otherwise, the default data is returned.
 *   - If you specified data with `setChain` it will return that.
 *
 * Summarizing the above, the prioritization hierarchy is `setChain` -> `foundry
    ↪ .toml` -> environment variable -> defaults.
 */
abstract contract StdChains {
    VmSafe private constant vm = VmSafe(address(uint160(uint256(keccak256("hevm
       ↪ cheat code")))));

    bool private initialized;

    // Maps from the chain's alias (matching the alias in the `foundry.toml`
        ↪ file) to chain data.
    mapping(string => Chain) private chains;
    // Maps from the chain's alias to it's default RPC URL.
    mapping(string => string) private defaultRpcUrls;
    // Maps from a chain ID to it's alias.
    mapping(uint256 => string) private idToAlias;
}
```

8.11.1 struct StdChains.ChainData

```
    struct ChainData {
        string name;
        uint256 chainId;
        string rpcUrl;
    }
```

8.11.2 struct StdChains.Chain

```
    struct Chain {
        // The chain name.
        string name;
        // The chain's Chain ID.
        uint256 chainId;
        // The chain's alias. (i.e. what gets specified in `foundry.toml`).
        string chainAlias;
        // A default RPC endpoint for this chain.
        // NOTE: This default RPC URL is included for convenience to facilitate
            ↪ quick tests and
        // experimentation. Do not use this RPC URL for production test suites,
            ↪ CI, or other heavy
        // usage as you will be throttled and this is a disservice to others who
            ↪  need this endpoint.
        string rpcUrl;
    }
```

8.11.3 struct StdCheatsSafe.RawTx1559

```
    // Data structures to parse Transaction objects from the broadcast artifact
    // that conform to EIP1559. The Raw structs is what is parsed from the JSON
    // and then converted to the one that is used by the user for better UX.

    struct RawTx1559 {
        string[] arguments;
        address contractAddress;
        string contractName;
        // json value name = function
        string functionSig;
        bytes32 hash;
        // json value name = tx
        RawTx1559Detail txDetail;
        // json value name = type
        string opcode;
    }
```

8.11.4 struct StdCheatsSafe.RawTx1559Detail

```
    struct RawTx1559Detail {
        AccessList[] accessList;
        bytes data;
        address from;
        bytes gas;
        bytes nonce;
        address to;
        bytes txType;
        bytes value;
    }
```

8.11.5 struct StdCheatsSafe.Tx1559

```
    struct Tx1559 {
        string[] arguments;
        address contractAddress;
        string contractName;
        string functionSig;
        bytes32 hash;
        Tx1559Detail txDetail;
        string opcode;
    }
```

8.11.6 struct StdCheatsSafe.Tx1559Detail

```
    struct Tx1559Detail {
        AccessList[] accessList;
        bytes data;
        address from;
        uint256 gas;
        uint256 nonce;
        address to;
        uint256 txType;
        uint256 value;
    }
```

8.11.7 struct StdCheatsSafe.TxLegacy

```
    // Data structures to parse Transaction objects from the broadcast artifact
    // that DO NOT conform to EIP1559. The Raw structs is what is parsed from
       ↪ the JSON
    // and then converted to the one that is used by the user for better UX.

    struct TxLegacy {
        string[] arguments;
        address contractAddress;
        string contractName;
        string functionSig;
        string hash;
        string opcode;
        TxDetailLegacy transaction;
    }
```

8.11.8 struct StdCheatsSafe.TxDetailLegacy

```
    struct TxDetailLegacy {
        AccessList[] accessList;
        uint256 chainId;
        bytes data;
        address from;
        uint256 gas;
        uint256 gasPrice;
        bytes32 hash;
        uint256 nonce;
        bytes1 opcode;
        bytes32 r;
        bytes32 s;
        uint256 txType;
        address to;
        uint8 v;
        uint256 value;
    }
```

8.11.9 struct StdCheatsSafe.AccessList

```
    struct AccessList {
        address accessAddress;
        bytes32[] storageKeys;
    }
```

8.11.10 struct StdCheatsSafe.RawReceipt

```
    // Data structures to parse Receipt objects from the broadcast artifact.
    // The Raw structs is what is parsed from the JSON
    // and then converted to the one that is used by the user for better UX.

    struct RawReceipt {
        bytes32 blockHash;
        bytes blockNumber;
        address contractAddress;
        bytes cumulativeGasUsed;
        bytes effectiveGasPrice;
        address from;
        bytes gasUsed;
        RawReceiptLog[] logs;
        bytes logsBloom;
        bytes status;
        address to;
        bytes32 transactionHash;
        bytes transactionIndex;
    }
```

8.11.11 struct StdCheatsSafe.Receipt

```
    struct Receipt {
        bytes32 blockHash;
        uint256 blockNumber;
        address contractAddress;
        uint256 cumulativeGasUsed;
        uint256 effectiveGasPrice;
        address from;
        uint256 gasUsed;
        ReceiptLog[] logs;
        bytes logsBloom;
        uint256 status;
        address to;
        bytes32 transactionHash;
        uint256 transactionIndex;
    }
```

8.11.12 struct StdCheatsSafe.EIP1559ScriptArtifact

```
    // Data structures to parse the entire broadcast artifact, assuming the
    // transactions conform to EIP1559.

    struct EIP1559ScriptArtifact {
        string[] libraries;
        string path;
        string[] pending;
        Receipt[] receipts;
        uint256 timestamp;
        Tx1559[] transactions;
        TxReturn[] txReturns;
    }
```

8.11.13 struct StdCheatsSafe.RawEIP1559ScriptArtifact

```
    struct RawEIP1559ScriptArtifact {
        string[] libraries;
        string path;
        string[] pending;
        RawReceipt[] receipts;
        TxReturn[] txReturns;
        uint256 timestamp;
        RawTx1559[] transactions;
    }
```

8.11.14 struct StdCheatsSafe.RawReceiptLog

```
    struct RawReceiptLog {
        // json value = address
        address logAddress;
        bytes32 blockHash;
        bytes blockNumber;
        bytes data;
        bytes logIndex;
        bool removed;
        bytes32[] topics;
        bytes32 transactionHash;
        bytes transactionIndex;
        bytes transactionLogIndex;
    }
```

8.11.15 struct StdCheatsSafe.ReceiptLog

```
    struct ReceiptLog {
        // json value = address
        address logAddress;
        bytes32 blockHash;
        uint256 blockNumber;
        bytes data;
        uint256 logIndex;
        bytes32[] topics;
        uint256 transactionIndex;
        uint256 transactionLogIndex;
        bool removed;
    }
```

8.11.16 struct StdCheatsSafe.TxReturn

```
    struct TxReturn {
        string internalType;
        string value;
    }
```

8.11.17 modifier skipWhenForking() [StdCheatsSafe]

```
    modifier skipWhenForking() {
        if (!isFork()) {
            _;
        }
    }
```

8.11.18 modifier skipWhenNotForking() [StdCheatsSafe]

```
    modifier skipWhenNotForking() {
        if (isFork()) {
            _;
        }
    }
```

8.11.19 modifier noGasMetering() [StdCheatsSafe]

```
    modifier noGasMetering() {
        vm.pauseGasMetering();
        // To prevent turning gas monitoring back on with nested functions that
            ↪ use this modifier,
        // we check if gasMetering started in the off position. If it did, we
            ↪ don't want to turn
        // it back on until we exit the top level function that used the
            ↪ modifier
        //
        // i.e. funcA() noGasMetering { funcB() }, where funcB has noGasMetering
            ↪  as well.
        // funcA will have `gasStartedOff` as false, funcB will have it as true,
        // so we only turn metering back on at the end of the funcA
        bool gasStartedOff = gasMeteringOff;
        gasMeteringOff = true;

        _;

        // if gas metering was on when this modifier was called, turn it back on
            ↪  at the end
        if (!gasStartedOff) {
            gasMeteringOff = false;
            vm.resumeGasMetering();
        }
    }
```

8.11.20 _bound(x, min, max) [StdUtils]

```
    function _bound(uint256 x, uint256 min, uint256 max) internal pure virtual
        ↪ returns (uint256 result) {
        require(min <= max, "StdUtils bound(uint256,uint256,uint256): Max is
            ↪ less than min.");
        // If x is between min and max, return x directly. This is to ensure
            ↪ that dictionary values
        // do not get shifted if the min is nonzero. More info: https://github.
            ↪ com/foundry-rs/forge-std/issues/188
        if (x >= min && x <= max) return x;

        uint256 size = max - min + 1;

        // If the value is 0, 1, 2, 3, warp that to min, min+1, min+2, min+3.
            ↪ Similarly for the UINT256_MAX side.
        // This helps ensure coverage of the min/max values.
        if (x <= 3 && size > x) return min + x;
        if (x >= UINT256_MAX - 3 && size > UINT256_MAX - x) return max - (
            ↪ UINT256_MAX - x);

        // Otherwise, wrap x into the range [min, max], i.e. the range is
            ↪ inclusive.
        if (x > max) {
            uint256 diff = x - max;
            uint256 rem = diff % size;
            if (rem == 0) return max;
            result = min + rem - 1;
        } else if (x < min) {
            uint256 diff = min - x;
            uint256 rem = diff % size;
            if (rem == 0) return min;
            result = max - rem + 1;
        }
    }
```

8.11.21 bound(x, min, max) [StdUtils]

```
    function bound(uint256 x, uint256 min, uint256 max) internal view virtual
        ↪ returns (uint256 result) {
        result = _bound(x, min, max);
        console2_log("Bound Result", result);
    }
```

8.11.22 bound(x, min, max) [StdUtils]

```
    function bound(int256 x, int256 min, int256 max) internal view virtual
        ↪ returns (int256 result) {
        require(min <= max, "StdUtils bound(int256,int256,int256): Max is less
            ↪ than min.");

        // Shifting all int256 values to uint256 to use _bound function. The
            ↪ range of two types are:
        // int256 : -(2**255) ~ (2**255 - 1)
        // uint256:      0      ~ (2**256 - 1)
        // So, add 2**255, INT256_MIN_ABS to the integer values.
        //
        // If the given integer value is -2**255, we cannot use `-uint256(-x)`
            ↪ because of the overflow.
        // So, use `~uint256(x) + 1` instead.
        uint256 _x = x < 0 ? (INT256_MIN_ABS - ~uint256(x) - 1) : (uint256(x) +
            ↪ INT256_MIN_ABS);
        uint256 _min = min < 0 ? (INT256_MIN_ABS - ~uint256(min) - 1) : (uint256
            ↪ (min) + INT256_MIN_ABS);
        uint256 _max = max < 0 ? (INT256_MIN_ABS - ~uint256(max) - 1) : (uint256
            ↪ (max) + INT256_MIN_ABS);

        uint256 y = _bound(_x, _min, _max);

        // To move it back to int256 value, subtract INT256_MIN_ABS at here.
        result = y < INT256_MIN_ABS ? int256(~(INT256_MIN_ABS - y) + 1) : int256
            ↪ (y - INT256_MIN_ABS);
        console2_log("Bound result", vm.toString(result));
    }
```

8.11.23 computeCreateAddress(deployer, nonce) [StdUtils]

```
    /// @dev Compute the address a contract will be deployed at for a given
        ↪ deployer address and nonce
    /// @notice adapated from Solmate implementation (https://github.com/Rari-
        ↪ Capital/solmate/blob/main/src/utils/LibRLP.sol)
    function computeCreateAddress(address deployer, uint256 nonce) internal pure
        ↪  virtual returns (address) {
        // forgefmt: disable-start
        // The integer zero is treated as an empty byte string, and as a result
            ↪ it only has a length prefix, 0x80, computed via 0x80 + 0.
        // A one byte integer uses its own value as its length prefix, there is
            ↪ no additional "0x80 + length" prefix that comes before it.
        if (nonce == 0x00)      return addressFromLast20Bytes(keccak256(abi.
            ↪ encodePacked(bytes1(0xd6), bytes1(0x94), deployer, bytes1(0x80)))
            ↪ );
        if (nonce <= 0x7f)      return addressFromLast20Bytes(keccak256(abi.
            ↪ encodePacked(bytes1(0xd6), bytes1(0x94), deployer, uint8(nonce)))
            ↪ );
        // Nonces greater than 1 byte all follow a consistent encoding scheme,
            ↪ where each value is preceded by a prefix of 0x80 + length.
        if (nonce <= 2**8 - 1)  return addressFromLast20Bytes(keccak256(abi.
            ↪ encodePacked(bytes1(0xd7), bytes1(0x94), deployer, bytes1(0x81),
            ↪ uint8(nonce))));
        if (nonce <= 2**16 - 1) return addressFromLast20Bytes(keccak256(abi.
            ↪ encodePacked(bytes1(0xd8), bytes1(0x94), deployer, bytes1(0x82),
            ↪ uint16(nonce))));
        if (nonce <= 2**24 - 1) return addressFromLast20Bytes(keccak256(abi.
            ↪ encodePacked(bytes1(0xd9), bytes1(0x94), deployer, bytes1(0x83),
            ↪ uint24(nonce))));
        // forgefmt: disable-end

        // More details about RLP encoding can be found here: https://eth.wiki/
            ↪ fundamentals/rlp
        // 0xda = 0xc0 (short RLP prefix) + 0x16 (length of: 0x94 ++ proxy ++ 0
            ↪ x84 ++ nonce)
        // 0x94 = 0x80 + 0x14 (0x14 = the length of an address, 20 bytes, in hex
            ↪ )
        // 0x84 = 0x80 + 0x04 (0x04 = the bytes length of the nonce, 4 bytes, in
            ↪  hex)
        // We assume nobody can have a nonce large enough to require more than
            ↪ 32 bytes.
        return addressFromLast20Bytes(
            keccak256(abi.encodePacked(bytes1(0xda), bytes1(0x94), deployer,
                ↪ bytes1(0x84), uint32(nonce)))
        );
    }
```

8.11.24 computeCreate2Address(salt, initcodeHash, deployer) [StdUtils]

```
    function computeCreate2Address(bytes32 salt, bytes32 initcodeHash, address
        ↪ deployer)
        internal
        pure
        virtual
        returns (address)
    {
        return addressFromLast20Bytes(keccak256(abi.encodePacked(bytes1(0xff),
            ↪ deployer, salt, initcodeHash)));
    }
```

8.11.25 bytesToUint(b) [StdUtils]

```
    function bytesToUint(bytes memory b) internal pure virtual returns (uint256)
        ↪  {
        require(b.length <= 32, "StdUtils bytesToUint(bytes): Bytes length
            ↪ exceeds 32.");
        return abi.decode(abi.encodePacked(new bytes(32 - b.length), b), (
            ↪ uint256));
    }
```

8.11.26 addressFromLast20Bytes(bytesValue) [StdUtils]

```
    function addressFromLast20Bytes(bytes32 bytesValue) private pure returns (
        ↪ address) {
        return address(uint160(uint256(bytesValue)));
    }
```

8.11.27 console2_log(p0, p1) [StdUtils]

```
    // Used to prevent the compilation of console, which shortens the
       ↪ compilation time when console is not used elsewhere.

    function console2_log(string memory p0, uint256 p1) private view {
        (bool status,) = address(CONSOLE2_ADDRESS).staticcall(abi.
           ↪ encodeWithSignature("log(string,uint256)", p0, p1));
        status;
    }
```

8.11.28 console2_log(p0, p1) [StdUtils]

```
    function console2_log(string memory p0, string memory p1) private view {
        (bool status,) = address(CONSOLE2_ADDRESS).staticcall(abi.
           ↪ encodeWithSignature("log(string,string)", p0, p1));
        status;
    }
```

8.11.29 assumeNoPrecompiles(addr) [StdCheatsSafe]

```
    function assumeNoPrecompiles(address addr) internal virtual {
        // Assembly required since `block.chainid` was introduced in 0.8.0.
        uint256 chainId;
        assembly {
            chainId := chainid()
        }
        assumeNoPrecompiles(addr, chainId);
    }
```

8.11.30 assumeNoPrecompiles(addr, chainId) [StdCheatsSafe]

```
    function assumeNoPrecompiles(address addr, uint256 chainId) internal pure
       ↪ virtual {
        // Note: For some chains like Optimism these are technically predeploys
           ↪ (i.e. bytecode placed at a specific
        // address), but the same rationale for excluding them applies so we
           ↪ include those too.

        // These should be present on all EVM-compatible chains.
        vm.assume(addr < address(0x1) || addr > address(0x9));

        // forgefmt: disable-start
        if (chainId == 10 || chainId == 420) {
            // https://github.com/ethereum-optimism/optimism/blob/
               ↪ eaa371a0184b56b7ca6d9eb9cb0a2b78b2ccd864/op-bindings/
               ↪ predeploys/addresses.go#L6-L21
            vm.assume(addr < address(0x4200000000000000000000000000000000000000)
               ↪  || addr > address(0x4200000000000000000000000000000000000800
               ↪ ));
        } else if (chainId == 42161 || chainId == 421613) {
            // https://developer.arbitrum.io/useful-addresses#arbitrum-
               ↪ precompiles-l2-same-on-all-arb-chains
            vm.assume(addr < address(0x0000000000000000000000000000000000000064)
               ↪  || addr > address(0x0000000000000000000000000000000000000068
               ↪ ));
        } else if (chainId == 43114 || chainId == 43113) {
            // https://github.com/ava-labs/subnet-evm/blob/47
               ↪ c03fd007ecaa6de2c52ea081596e0a88401f58/precompile/params.go#
               ↪ L18-L59
            vm.assume(addr < address(0x0100000000000000000000000000000000000000)
               ↪  || addr > address(0x01000000000000000000000000000000000000ff
               ↪ ));
            vm.assume(addr < address(0x0200000000000000000000000000000000000000)
               ↪  || addr > address(0x02000000000000000000000000000000000000FF
               ↪ ));
            vm.assume(addr < address(0x0300000000000000000000000000000000000000)
                ↪ || addr > address(0x03000000000000000000000000000000000000Ff
                ↪ ));
        }
        // forgefmt: disable-end
    }
```

8.11.31 readEIP1559ScriptArtifact(path) [StdCheatsSafe]

```
    function readEIP1559ScriptArtifact(string memory path)
        internal
        view
        virtual
        returns (EIP1559ScriptArtifact memory)
    {
        string memory data = vm.readFile(path);
        bytes memory parsedData = vm.parseJson(data);
        RawEIP1559ScriptArtifact memory rawArtifact = abi.decode(parsedData, (
            ↪ RawEIP1559ScriptArtifact));
        EIP1559ScriptArtifact memory artifact;
        artifact.libraries = rawArtifact.libraries;
        artifact.path = rawArtifact.path;
        artifact.timestamp = rawArtifact.timestamp;
        artifact.pending = rawArtifact.pending;
        artifact.txReturns = rawArtifact.txReturns;
        artifact.receipts = rawToConvertedReceipts(rawArtifact.receipts);
        artifact.transactions = rawToConvertedEIPTx1559s(rawArtifact.
            ↪ transactions);
        return artifact;
    }
```

8.11.32 rawToConvertedEIPTx1559s(rawTxs) [StdCheatsSafe]

```
    function rawToConvertedEIPTx1559s(RawTx1559[] memory rawTxs) internal pure
        ↪ virtual returns (Tx1559[] memory) {
        Tx1559[] memory txs = new Tx1559[](rawTxs.length);
        for (uint256 i; i < rawTxs.length; i++) {
            txs[i] = rawToConvertedEIPTx1559(rawTxs[i]);
        }
        return txs;
    }
```

8.11.33 rawToConvertedEIPTx1559(rawTx) [StdCheatsSafe]

```
    function rawToConvertedEIPTx1559(RawTx1559 memory rawTx) internal pure
        ↪ virtual returns (Tx1559 memory) {
        Tx1559 memory transaction;
        transaction.arguments = rawTx.arguments;
        transaction.contractName = rawTx.contractName;
        transaction.functionSig = rawTx.functionSig;
        transaction.hash = rawTx.hash;
        transaction.txDetail = rawToConvertedEIP1559Detail(rawTx.txDetail);
        transaction.opcode = rawTx.opcode;
        return transaction;
    }
```

8.11.34 rawToConvertedEIP1559Detail(rawDetail) [StdCheatsSafe]

```
    function rawToConvertedEIP1559Detail(RawTx1559Detail memory rawDetail)
        internal
        pure
        virtual
        returns (Tx1559Detail memory)
    {
        Tx1559Detail memory txDetail;
        txDetail.data = rawDetail.data;
        txDetail.from = rawDetail.from;
        txDetail.to = rawDetail.to;
        txDetail.nonce = _bytesToUint(rawDetail.nonce);
        txDetail.txType = _bytesToUint(rawDetail.txType);
        txDetail.value = _bytesToUint(rawDetail.value);
        txDetail.gas = _bytesToUint(rawDetail.gas);
        txDetail.accessList = rawDetail.accessList;
        return txDetail;
    }
```

8.11.35 readTx1559s(path) [StdCheatsSafe]

```
    function readTx1559s(string memory path) internal view virtual returns (
        ↪ Tx1559[] memory) {
        string memory deployData = vm.readFile(path);
        bytes memory parsedDeployData = vm.parseJson(deployData, ".transactions"
            ↪ );
        RawTx1559[] memory rawTxs = abi.decode(parsedDeployData, (RawTx1559[]));
        return rawToConvertedEIPTx1559s(rawTxs);
    }
```

8.11.36 readTx1559(path, index) [StdCheatsSafe]

```
    function readTx1559(string memory path, uint256 index) internal view virtual
        ↪  returns (Tx1559 memory) {
        string memory deployData = vm.readFile(path);
        string memory key = string(abi.encodePacked(".transactions[", vm.
            ↪ toString(index), "]"));
        bytes memory parsedDeployData = vm.parseJson(deployData, key);
        RawTx1559 memory rawTx = abi.decode(parsedDeployData, (RawTx1559));
        return rawToConvertedEIPTx1559(rawTx);
    }
```

8.11.37 readReceipts(path) [StdCheatsSafe]

```
    // Analogous to readTransactions, but for receipts.
    function readReceipts(string memory path) internal view virtual returns (
        ↪ Receipt[] memory) {
        string memory deployData = vm.readFile(path);
        bytes memory parsedDeployData = vm.parseJson(deployData, ".receipts");
        RawReceipt[] memory rawReceipts = abi.decode(parsedDeployData, (
            ↪ RawReceipt[]));
        return rawToConvertedReceipts(rawReceipts);
    }
```

8.11.38 readReceipt(path, index) [StdCheatsSafe]

```
    function readReceipt(string memory path, uint256 index) internal view
        ↪ virtual returns (Receipt memory) {
        string memory deployData = vm.readFile(path);
        string memory key = string(abi.encodePacked(".receipts[", vm.toString(
            ↪ index), "]"));
        bytes memory parsedDeployData = vm.parseJson(deployData, key);
        RawReceipt memory rawReceipt = abi.decode(parsedDeployData, (RawReceipt)
            ↪ );
        return rawToConvertedReceipt(rawReceipt);
    }
```

8.11.39 rawToConvertedReceipts(rawReceipts) [StdCheatsSafe]

```
function rawToConvertedReceipts(RawReceipt[] memory rawReceipts) internal
    ↪ pure virtual returns (Receipt[] memory) {
    Receipt[] memory receipts = new Receipt[](rawReceipts.length);
    for (uint256 i; i < rawReceipts.length; i++) {
        receipts[i] = rawToConvertedReceipt(rawReceipts[i]);
    }
    return receipts;
}
```

8.11.40 rawToConvertedReceipt(rawReceipt) [StdCheatsSafe]

```
function rawToConvertedReceipt(RawReceipt memory rawReceipt) internal pure
    ↪ virtual returns (Receipt memory) {
    Receipt memory receipt;
    receipt.blockHash = rawReceipt.blockHash;
    receipt.to = rawReceipt.to;
    receipt.from = rawReceipt.from;
    receipt.contractAddress = rawReceipt.contractAddress;
    receipt.effectiveGasPrice = _bytesToUint(rawReceipt.effectiveGasPrice);
    receipt.cumulativeGasUsed = _bytesToUint(rawReceipt.cumulativeGasUsed);
    receipt.gasUsed = _bytesToUint(rawReceipt.gasUsed);
    receipt.status = _bytesToUint(rawReceipt.status);
    receipt.transactionIndex = _bytesToUint(rawReceipt.transactionIndex);
    receipt.blockNumber = _bytesToUint(rawReceipt.blockNumber);
    receipt.logs = rawToConvertedReceiptLogs(rawReceipt.logs);
    receipt.logsBloom = rawReceipt.logsBloom;
    receipt.transactionHash = rawReceipt.transactionHash;
    return receipt;
}
```

8.11.41 rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe]

```
function rawToConvertedReceiptLogs(RawReceiptLog[] memory rawLogs)
    internal
    pure
    virtual
    returns (ReceiptLog[] memory)
{
    ReceiptLog[] memory logs = new ReceiptLog[](rawLogs.length);
    for (uint256 i; i < rawLogs.length; i++) {
        logs[i].logAddress = rawLogs[i].logAddress;
        logs[i].blockHash = rawLogs[i].blockHash;
        logs[i].blockNumber = _bytesToUint(rawLogs[i].blockNumber);
        logs[i].data = rawLogs[i].data;
        logs[i].logIndex = _bytesToUint(rawLogs[i].logIndex);
        logs[i].topics = rawLogs[i].topics;
        logs[i].transactionIndex = _bytesToUint(rawLogs[i].transactionIndex)
            ↪ ;
        logs[i].transactionLogIndex = _bytesToUint(rawLogs[i].
            ↪ transactionLogIndex);
        logs[i].removed = rawLogs[i].removed;
    }
    return logs;
}
```

8.11.42 deployCode(what, args) [StdCheatsSafe]

```
// Deploy a contract by fetching the contract bytecode from
// the artifacts directory
// e.g. `deployCode(code, abi.encode(arg1,arg2,arg3))`
function deployCode(string memory what, bytes memory args) internal virtual
    ↪ returns (address addr) {
        bytes memory bytecode = abi.encodePacked(vm.getCode(what), args);
        /// @solidity memory-safe-assembly
        assembly {
            addr := create(0, add(bytecode, 0x20), mload(bytecode))
        }

        require(addr != address(0), "StdCheats deployCode(string,bytes):
            ↪ Deployment failed.");
    }
```

8.11.43 deployCode(what) [StdCheatsSafe]

```
    function deployCode(string memory what) internal virtual returns (address
        ↪ addr) {
        bytes memory bytecode = vm.getCode(what);
        /// @solidity memory-safe-assembly
        assembly {
            addr := create(0, add(bytecode, 0x20), mload(bytecode))
        }

        require(addr != address(0), "StdCheats deployCode(string): Deployment
            ↪ failed.");
    }
```

8.11.44 deployCode(what, args, val) [StdCheatsSafe]

```
    /// @dev deploy contract with value on construction
    function deployCode(string memory what, bytes memory args, uint256 val)
        ↪ internal virtual returns (address addr) {
        bytes memory bytecode = abi.encodePacked(vm.getCode(what), args);
        /// @solidity memory-safe-assembly
        assembly {
            addr := create(val, add(bytecode, 0x20), mload(bytecode))
        }

        require(addr != address(0), "StdCheats deployCode(string,bytes,uint256):
            ↪  Deployment failed.");
    }
```

8.11.45 deployCode(what, val) [StdCheatsSafe]

```
    function deployCode(string memory what, uint256 val) internal virtual
        ↪ returns (address addr) {
        bytes memory bytecode = vm.getCode(what);
        /// @solidity memory-safe-assembly
        assembly {
            addr := create(val, add(bytecode, 0x20), mload(bytecode))
        }

        require(addr != address(0), "StdCheats deployCode(string,uint256):
            ↪ Deployment failed.");
    }
```

8.11.46 makeAddrAndKey(name) [StdCheatsSafe]

```
    // creates a labeled address and the corresponding private key
    function makeAddrAndKey(string memory name) internal virtual returns (
        ↪ address addr, uint256 privateKey) {
        privateKey = uint256(keccak256(abi.encodePacked(name)));
        addr = vm.addr(privateKey);
        vm.label(addr, name);
    }
```

8.11.47 makeAddr(name) [StdCheatsSafe]

```
    // creates a labeled address
    function makeAddr(string memory name) internal virtual returns (address addr
        ↪ ) {
        (addr,) = makeAddrAndKey(name);
    }
```

8.11.48 deriveRememberKey(mnemonic, index) [StdCheatsSafe]

```
    function deriveRememberKey(string memory mnemonic, uint32 index)
        internal
        virtual
        returns (address who, uint256 privateKey)
    {
        privateKey = vm.deriveKey(mnemonic, index);
        who = vm.rememberKey(privateKey);
    }
```

8.11.49 _bytesToUint(b) [StdCheatsSafe]

```
    function _bytesToUint(bytes memory b) private pure returns (uint256) {
        require(b.length <= 32, "StdCheats _bytesToUint(bytes): Bytes length
            ↪ exceeds 32.");
        return abi.decode(abi.encodePacked(new bytes(32 - b.length), b), (
            ↪ uint256));
    }
```

8.11.50 isFork() [StdCheatsSafe]

```
    function isFork() internal view virtual returns (bool status) {
        try vm.activeFork() {
            status = true;
        } catch (bytes memory) {}
    }
```

8.11.51 getChain(chainAlias) [StdChains]

```
    // The RPC URL will be fetched from config or defaultRpcUrls if possible.
    function getChain(string memory chainAlias) internal virtual returns (Chain
        ↪ memory chain) {
        require(bytes(chainAlias).length != 0, "StdChains getChain(string):
            ↪ Chain alias cannot be the empty string.");

        initialize();
        chain = chains[chainAlias];
        require(
            chain.chainId != 0,
            string(abi.encodePacked("StdChains getChain(string): Chain with
                ↪ alias \"", chainAlias, "\" not found."))
        );

        chain = getChainWithUpdatedRpcUrl(chainAlias, chain);
    }
```

8.11.52 getChain(chainId) [StdChains]

```
    function getChain(uint256 chainId) internal virtual returns (Chain memory
        ↪ chain) {
        require(chainId != 0, "StdChains getChain(uint256): Chain ID cannot be
            ↪ 0.");
        initialize();
        string memory chainAlias = idToAlias[chainId];

        chain = chains[chainAlias];

        require(
            chain.chainId != 0,
            string(abi.encodePacked("StdChains getChain(uint256): Chain with ID
                ↪ ", vm.toString(chainId), " not found."))
        );

        chain = getChainWithUpdatedRpcUrl(chainAlias, chain);
    }
```

8.11.53 setChain(chainAlias, chain) [StdChains]

```
    // set chain info, with priority to argument's rpcUrl field.
    function setChain(string memory chainAlias, ChainData memory chain) internal
        ↪ virtual {
        require(
            bytes(chainAlias).length != 0,
            "StdChains setChain(string,ChainData): Chain alias cannot be the
                ↪ empty string."
        );

        require(chain.chainId != 0, "StdChains setChain(string,ChainData): Chain
            ↪ ID cannot be 0.");

        initialize();
        string memory foundAlias = idToAlias[chain.chainId];

        require(
            bytes(foundAlias).length == 0 || keccak256(bytes(foundAlias)) ==
                ↪ keccak256(bytes(chainAlias)),
            string(
                abi.encodePacked(
                    "StdChains setChain(string,ChainData): Chain ID ",
                    vm.toString(chain.chainId),
                    " already used by \"",
                    foundAlias,
                    "\"."
                )
            )
        );

        uint256 oldChainId = chains[chainAlias].chainId;
        delete idToAlias[oldChainId];

        chains[chainAlias] =
            Chain({name: chain.name, chainId: chain.chainId, chainAlias:
                ↪ chainAlias, rpcUrl: chain.rpcUrl});
        idToAlias[chain.chainId] = chainAlias;
    }
```

8.11.54 setChain(chainAlias, chain) [StdChains]

```
    // set chain info, with priority to argument's rpcUrl field.
    function setChain(string memory chainAlias, Chain memory chain) internal
        ↪ virtual {
        setChain(chainAlias, ChainData({name: chain.name, chainId: chain.chainId
            ↪ , rpcUrl: chain.rpcUrl}));
    }
```

8.11.55 _toUpper(str) [StdChains]

```
function _toUpper(string memory str) private pure returns (string memory) {
    bytes memory strb = bytes(str);
    bytes memory copy = new bytes(strb.length);
    for (uint256 i = 0; i < strb.length; i++) {
        bytes1 b = strb[i];
        if (b >= 0x61 && b <= 0x7A) {
            copy[i] = bytes1(uint8(b) - 32);
        } else {
            copy[i] = b;
        }
    }
    return string(copy);
}
```

8.11.56 getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains]

```
// lookup rpcUrl, in descending order of priority:
// current -> config (foundry.toml) -> environment variable -> default
function getChainWithUpdatedRpcUrl(string memory chainAlias, Chain memory
   ↪ chain) private returns (Chain memory) {
    if (bytes(chain.rpcUrl).length == 0) {
        try vm.rpcUrl(chainAlias) returns (string memory configRpcUrl) {
            chain.rpcUrl = configRpcUrl;
        } catch (bytes memory err) {
            chain.rpcUrl =
                vm.envOr(string(abi.encodePacked(_toUpper(chainAlias), "
                    ↪ _RPC_URL")), defaultRpcUrls[chainAlias]);
            // distinguish 'not found' from 'cannot read'
            bytes memory notFoundError =
                abi.encodeWithSignature("CheatCodeError", string(abi.
                    ↪ encodePacked("invalid rpc url ", chainAlias)));
            if (keccak256(notFoundError) != keccak256(err) || bytes(chain.
                ↪ rpcUrl).length == 0) {
                /// @solidity memory-safe-assembly
                assembly {
                    revert(add(32, err), mload(err))
                }
            }
        }
    }
    return chain;
}
```

8.11.57 initialize() [StdChains]

```
function initialize() private {
    if (initialized) return;

    initialized = true;

    // If adding an RPC here, make sure to test the default RPC URL in `
        ↪ testRpcs`
    setChainWithDefaultRpcUrl("anvil", ChainData("Anvil", 31337, "http
        ↪ ://127.0.0.1:8545"));
    setChainWithDefaultRpcUrl(
        "mainnet", ChainData("Mainnet", 1, "https://mainnet.infura.io/v3
            ↪ /6770454bc6ea42c58aac12978531b93f")
    );
    setChainWithDefaultRpcUrl(
        "goerli", ChainData("Goerli", 5, "https://goerli.infura.io/v3
            ↪ /6770454bc6ea42c58aac12978531b93f")
    );
    setChainWithDefaultRpcUrl(
        "sepolia", ChainData("Sepolia", 11155111, "https://sepolia.infura.io
            ↪ /v3/6770454bc6ea42c58aac12978531b93f")
    );
    setChainWithDefaultRpcUrl("optimism", ChainData("Optimism", 10, "https
        ↪ ://mainnet.optimism.io"));
    setChainWithDefaultRpcUrl("optimism_goerli", ChainData("Optimism Goerli"
        ↪ , 420, "https://goerli.optimism.io"));
    setChainWithDefaultRpcUrl("arbitrum_one", ChainData("Arbitrum One",
        ↪ 42161, "https://arb1.arbitrum.io/rpc"));
    setChainWithDefaultRpcUrl(
        "arbitrum_one_goerli", ChainData("Arbitrum One Goerli", 421613, "
            ↪ https://goerli-rollup.arbitrum.io/rpc")
    );
    setChainWithDefaultRpcUrl("arbitrum_nova", ChainData("Arbitrum Nova",
        ↪ 42170, "https://nova.arbitrum.io/rpc"));
    setChainWithDefaultRpcUrl("polygon", ChainData("Polygon", 137, "https://
        ↪ polygon-rpc.com"));
    setChainWithDefaultRpcUrl(
        "polygon_mumbai", ChainData("Polygon Mumbai", 80001, "https://rpc-
            ↪ mumbai.maticvigil.com")
    );
    setChainWithDefaultRpcUrl("avalanche", ChainData("Avalanche", 43114, "
        ↪ https://api.avax.network/ext/bc/C/rpc"));
    setChainWithDefaultRpcUrl(
        "avalanche_fuji", ChainData("Avalanche Fuji", 43113, "https://api.
            ↪ avax-test.network/ext/bc/C/rpc")
    );
    setChainWithDefaultRpcUrl(
        "bnb_smart_chain", ChainData("BNB Smart Chain", 56, "https://bsc-
            ↪ dataseed1.binance.org")
    );
    setChainWithDefaultRpcUrl(
        "bnb_smart_chain_testnet",
        ChainData("BNB Smart Chain Testnet", 97, "https://data-seed-prebsc
            ↪ -1-s1.binance.org:8545")
    );
    setChainWithDefaultRpcUrl("gnosis_chain", ChainData("Gnosis Chain", 100,
        ↪  "https://rpc.gnosischain.com"));
}
```

8.11.58 setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains]

```
    // set chain info, with priority to chainAlias' rpc url in foundry.toml
    function setChainWithDefaultRpcUrl(string memory chainAlias, ChainData
        ↪ memory chain) private {
        string memory rpcUrl = chain.rpcUrl;
        defaultRpcUrls[chainAlias] = rpcUrl;
        chain.rpcUrl = "";
        setChain(chainAlias, chain);
        chain.rpcUrl = rpcUrl; // restore argument
    }
```

8.11.59 run() X

```
    function run() external {
        config = ScriptTools.loadConfig();
        OptionalLoadDependencies lp = new OptionalLoadDependencies();   // Try
            ↪ catch needs external function
        try lp.loadDependencies("core") returns (string memory deps) {
            dependencies = deps;
        } catch {
            // Fallback to chainlog
        }
        dss = MCD.loadFromChainlog(config.readAddress("chainlog"));

        d3mType = config.readString("type");
        admin = config.readAddress("admin");
        hub = dependencies.eq("") ? dss.chainlog.getAddress("DIRECT_HUB") :
            ↪ dependencies.readAddress("hub");
        ilk = config.readString("ilk").stringToBytes32();

        vm.startBroadcast();
        if (d3mType.eq("aave")) {
            d3m = D3MDeploy.deployAave(
                msg.sender,
                admin,
                ilk,
                address(dss.vat),
                hub,
                address(dss.dai),
                config.readAddress("lendingPool")
            );
        } else if (d3mType.eq("compound")) {
            d3m = D3MDeploy.deployCompound(
                msg.sender,
                admin,
                ilk,
                address(dss.vat),
                hub,
                config.readAddress("cdai")
            );
        } else {
            revert("unknown-d3m-type");
        }
        vm.stopBroadcast();

        ScriptTools.exportContract("pool", d3m.pool);
        ScriptTools.exportContract("plan", d3m.plan);
        ScriptTools.exportContract("oracle", d3m.oracle);
    }
```

8.12 contract D3MInitScript

```
contract D3MInitScript is Script {

    using stdJson for string;
    using ScriptTools for string;

    uint256 constant BPS = 10 ** 4;
    uint256 constant RAY = 10 ** 27;
    uint256 constant RAD = 10 ** 45;

    string config;
    string dependencies;
    DssInstance dss;

    string d3mType;
    bytes32 ilk;
    D3MInstance d3m;
    D3MCommonConfig cfg;
    D3MAaveConfig aaveCfg;
    D3MCompoundConfig compoundCfg;

}
```

Inherited:

```
// ?? SCRIPT
abstract contract Script is StdChains, StdCheatsSafe, StdUtils, ScriptBase {
    // Note: IS_SCRIPT() must return true.
    bool public IS_SCRIPT = true;
}
```

```
abstract contract ScriptBase is CommonBase {
    // Used when deploying with create2, https://github.com/Arachnid/
        ↪ deterministic-deployment-proxy.
    address internal constant CREATE2_FACTORY = 0
        ↪ x4e59b44847b379578588920cA78FbF26c0B4956C;

    VmSafe internal constant vmSafe = VmSafe(VM_ADDRESS);
}
```

```
abstract contract CommonBase {
    // Cheat code address, 0x7109709ECfa91a80626fF3989D68f67F5b1DD12D.
    address internal constant VM_ADDRESS = address(uint160(uint256(keccak256("
        ↪ hevm cheat code"))));
    // console.sol and console2.sol work by executing a staticcall to this
        ↪ address.
    address internal constant CONSOLE = 0
        ↪ x000000000000000000636F6e736F6c652e6c6f67;
    // Default address for tx.origin and msg.sender, 0
        ↪ x1804c8AB1F12E6bbf3894d4083f33e07309d1f38.
    address internal constant DEFAULT_SENDER = address(uint160(uint256(keccak256
        ↪ ("foundry default caller"))));
    // Address of the test contract, deployed by the DEFAULT_SENDER.
    address internal constant DEFAULT_TEST_CONTRACT = 0
        ↪ x5615dEB798BB3E4dFa0139dFa1b3D433Cc23b72f;
    // Deterministic deployment address of the Multicall3 contract.
    address internal constant MULTICALL3_ADDRESS = 0
        ↪ xcA11bde05977b3631167028862bE2a173976CA11;

    uint256 internal constant UINT256_MAX =
        115792089237316195423570985008687907853269984665640564039457584007913129639935
            ↪

    Vm internal constant vm = Vm(VM_ADDRESS);
    StdStorage internal stdstore;
}
```

```
abstract contract StdUtils {
    VmSafe private constant vm = VmSafe(address(uint160(uint256(keccak256("hevm
       ↪ cheat code")))));
    address private constant CONSOLE2_ADDRESS = 0
       ↪ x000000000000000000636F6e736F6c652e6c6f67;

    uint256 private constant INT256_MIN_ABS =
        57896044618658097711785492504343953926634992332820282019728792003956564819968;
           ↪
    uint256 private constant UINT256_MAX =
        115792089237316195423570985008687907853269984665640564039457584007913129639935;
           ↪
}
```

```
abstract contract StdCheatsSafe {
    Vm private constant vm = Vm(address(uint160(uint256(keccak256("hevm cheat
       ↪ code")))));

    bool private gasMeteringOff;
}
```

```
/**
 * StdChains provides information about EVM compatible chains that can be used
    ↪ in scripts/tests.
 * For each chain, the chain's name, chain ID, and a default RPC URL are
    ↪ provided. Chains are
 * identified by their alias, which is the same as the alias in the ‘[
    ↪ rpc_endpoints]‘ section of
 * the ‘foundry.toml‘ file. For best UX, ensure the alias in the ‘foundry.toml‘
    ↪ file match the
 * alias used in this contract, which can be found as the first argument to the
 * ‘setChainWithDefaultRpcUrl‘ call in the ‘initialize‘ function.
 *
 * There are two main ways to use this contract:
 *   1. Set a chain with ‘setChain(string memory chainAlias, ChainData memory
    ↪ chain)‘ or
 *      ‘setChain(string memory chainAlias, Chain memory chain)‘
 *   2. Get a chain with ‘getChain(string memory chainAlias)‘ or ‘getChain(
    ↪ uint256 chainId)‘.
 *
 * The first time either of those are used, chains are initialized with the
    ↪ default set of RPC URLs.
 * This is done in ‘initialize‘, which uses ‘setChainWithDefaultRpcUrl‘.
    ↪ Defaults are recorded in
 * ‘defaultRpcUrls‘.
 *
 * The ‘setChain‘ function is straightforward, and it simply saves off the given
    ↪  chain data.
 *
 * The ‘getChain‘ methods use ‘getChainWithUpdatedRpcUrl‘ to return a chain. For
    ↪  example, let's say
 * we want to retrieve ‘mainnet‘'s RPC URL:
 *   - If you haven't set any mainnet chain info with ‘setChain‘, you haven't
    ↪ specified that
 *     chain in ‘foundry.toml‘ and no env var is set, the default data and RPC
    ↪ URL will be returned.
 *   - If you have set a mainnet RPC URL in ‘foundry.toml‘ it will return that,
    ↪ if valid (e.g. if
 *     a URL is given or if an environment variable is given and that
    ↪ environment variable exists).
 *     Otherwise, the default data is returned.
 *   - If you specified data with ‘setChain‘ it will return that.
 *
 * Summarizing the above, the prioritization hierarchy is ‘setChain‘ -> ‘foundry
    ↪ .toml‘ -> environment variable -> defaults.
 */
abstract contract StdChains {
    VmSafe private constant vm = VmSafe(address(uint160(uint256(keccak256("hevm
        ↪ cheat code")))));

    bool private initialized;

    // Maps from the chain's alias (matching the alias in the `foundry.toml`
        ↪ file) to chain data.
    mapping(string => Chain) private chains;
    // Maps from the chain's alias to it's default RPC URL.
    mapping(string => string) private defaultRpcUrls;
    // Maps from a chain ID to it's alias.
    mapping(uint256 => string) private idToAlias;
}
```

8.12.1 struct StdChains.ChainData

```
    struct ChainData {
        string name;
        uint256 chainId;
        string rpcUrl;
    }
```

8.12.2 struct StdChains.Chain

```
    struct Chain {
        // The chain name.
        string name;
        // The chain's Chain ID.
        uint256 chainId;
        // The chain's alias. (i.e. what gets specified in `foundry.toml`).
        string chainAlias;
        // A default RPC endpoint for this chain.
        // NOTE: This default RPC URL is included for convenience to facilitate
            ↪ quick tests and
        // experimentation. Do not use this RPC URL for production test suites,
            ↪ CI, or other heavy
        // usage as you will be throttled and this is a disservice to others who
            ↪  need this endpoint.
        string rpcUrl;
    }
```

8.12.3 struct StdCheatsSafe.RawTx1559

```
    // Data structures to parse Transaction objects from the broadcast artifact
    // that conform to EIP1559. The Raw structs is what is parsed from the JSON
    // and then converted to the one that is used by the user for better UX.

    struct RawTx1559 {
        string[] arguments;
        address contractAddress;
        string contractName;
        // json value name = function
        string functionSig;
        bytes32 hash;
        // json value name = tx
        RawTx1559Detail txDetail;
        // json value name = type
        string opcode;
    }
```

8.12.4 struct StdCheatsSafe.RawTx1559Detail

```
    struct RawTx1559Detail {
        AccessList[] accessList;
        bytes data;
        address from;
        bytes gas;
        bytes nonce;
        address to;
        bytes txType;
        bytes value;
    }
```

8.12.5 struct StdCheatsSafe.Tx1559

```
    struct Tx1559 {
        string[] arguments;
        address contractAddress;
        string contractName;
        string functionSig;
        bytes32 hash;
        Tx1559Detail txDetail;
        string opcode;
    }
```

8.12.6 struct StdCheatsSafe.Tx1559Detail

```
    struct Tx1559Detail {
        AccessList[] accessList;
        bytes data;
        address from;
        uint256 gas;
        uint256 nonce;
        address to;
        uint256 txType;
        uint256 value;
    }
```

8.12.7 struct StdCheatsSafe.TxLegacy

```
    // Data structures to parse Transaction objects from the broadcast artifact
    // that DO NOT conform to EIP1559. The Raw structs is what is parsed from
      ↪ the JSON
    // and then converted to the one that is used by the user for better UX.

    struct TxLegacy {
        string[] arguments;
        address contractAddress;
        string contractName;
        string functionSig;
        string hash;
        string opcode;
        TxDetailLegacy transaction;
    }
```

8.12.8 struct StdCheatsSafe.TxDetailLegacy

```
    struct TxDetailLegacy {
        AccessList[] accessList;
        uint256 chainId;
        bytes data;
        address from;
        uint256 gas;
        uint256 gasPrice;
        bytes32 hash;
        uint256 nonce;
        bytes1 opcode;
        bytes32 r;
        bytes32 s;
        uint256 txType;
        address to;
        uint8 v;
        uint256 value;
    }
```

8.12.9 struct StdCheatsSafe.AccessList

```
    struct AccessList {
        address accessAddress;
        bytes32[] storageKeys;
    }
```

8.12.10 struct StdCheatsSafe.RawReceipt

```
    // Data structures to parse Receipt objects from the broadcast artifact.
    // The Raw structs is what is parsed from the JSON
    // and then converted to the one that is used by the user for better UX.

    struct RawReceipt {
        bytes32 blockHash;
        bytes blockNumber;
        address contractAddress;
        bytes cumulativeGasUsed;
        bytes effectiveGasPrice;
        address from;
        bytes gasUsed;
        RawReceiptLog[] logs;
        bytes logsBloom;
        bytes status;
        address to;
        bytes32 transactionHash;
        bytes transactionIndex;
    }
```

8.12.11 struct StdCheatsSafe.Receipt

```
    struct Receipt {
        bytes32 blockHash;
        uint256 blockNumber;
        address contractAddress;
        uint256 cumulativeGasUsed;
        uint256 effectiveGasPrice;
        address from;
        uint256 gasUsed;
        ReceiptLog[] logs;
        bytes logsBloom;
        uint256 status;
        address to;
        bytes32 transactionHash;
        uint256 transactionIndex;
    }
```

8.12.12 struct StdCheatsSafe.EIP1559ScriptArtifact

```
    // Data structures to parse the entire broadcast artifact, assuming the
    // transactions conform to EIP1559.
    struct EIP1559ScriptArtifact {
        string[] libraries;
        string path;
        string[] pending;
        Receipt[] receipts;
        uint256 timestamp;
        Tx1559[] transactions;
        TxReturn[] txReturns;
    }
```

8.12.13 struct StdCheatsSafe.RawEIP1559ScriptArtifact

```
    struct RawEIP1559ScriptArtifact {
        string[] libraries;
        string path;
        string[] pending;
        RawReceipt[] receipts;
        TxReturn[] txReturns;
        uint256 timestamp;
        RawTx1559[] transactions;
    }
```

8.12.14 struct StdCheatsSafe.RawReceiptLog

```
    struct RawReceiptLog {
        // json value = address
        address logAddress;
        bytes32 blockHash;
        bytes blockNumber;
        bytes data;
        bytes logIndex;
        bool removed;
        bytes32[] topics;
        bytes32 transactionHash;
        bytes transactionIndex;
        bytes transactionLogIndex;
    }
```

8.12.15 struct StdCheatsSafe.ReceiptLog

```
    struct ReceiptLog {
        // json value = address
        address logAddress;
        bytes32 blockHash;
        uint256 blockNumber;
        bytes data;
        uint256 logIndex;
        bytes32[] topics;
        uint256 transactionIndex;
        uint256 transactionLogIndex;
        bool removed;
    }
```

8.12.16 struct StdCheatsSafe.TxReturn

```
    struct TxReturn {
        string internalType;
        string value;
    }
```

8.12.17 modifier skipWhenForking() [StdCheatsSafe]

```
modifier skipWhenForking() {
    if (!isFork()) {
        _;
    }
}
```

8.12.18 modifier skipWhenNotForking() [StdCheatsSafe]

```
modifier skipWhenNotForking() {
    if (isFork()) {
        _;
    }
}
```

8.12.19 modifier noGasMetering() [StdCheatsSafe]

```
modifier noGasMetering() {
    vm.pauseGasMetering();
    // To prevent turning gas monitoring back on with nested functions that
        ↪ use this modifier,
    // we check if gasMetering started in the off position. If it did, we
        ↪ don't want to turn
    // it back on until we exit the top level function that used the
        ↪ modifier
    //
    // i.e. funcA() noGasMetering { funcB() }, where funcB has noGasMetering
        ↪  as well.
    // funcA will have `gasStartedOff` as false, funcB will have it as true,
    // so we only turn metering back on at the end of the funcA
    bool gasStartedOff = gasMeteringOff;
    gasMeteringOff = true;

    _;

    // if gas metering was on when this modifier was called, turn it back on
        ↪  at the end
    if (!gasStartedOff) {
        gasMeteringOff = false;
        vm.resumeGasMetering();
    }
}
```

8.12.20 _bound(x, min, max) [StdUtils]

```
function _bound(uint256 x, uint256 min, uint256 max) internal pure virtual
    ↪ returns (uint256 result) {
    require(min <= max, "StdUtils bound(uint256,uint256,uint256): Max is
        ↪ less than min.");
    // If x is between min and max, return x directly. This is to ensure
        ↪ that dictionary values
    // do not get shifted if the min is nonzero. More info: https://github.
        ↪ com/foundry-rs/forge-std/issues/188
    if (x >= min && x <= max) return x;

    uint256 size = max - min + 1;

    // If the value is 0, 1, 2, 3, warp that to min, min+1, min+2, min+3.
        ↪ Similarly for the UINT256_MAX side.
    // This helps ensure coverage of the min/max values.
    if (x <= 3 && size > x) return min + x;
    if (x >= UINT256_MAX - 3 && size > UINT256_MAX - x) return max - (
        ↪ UINT256_MAX - x);
        // Otherwise, wrap x into the range [min, max], i.e. the range is
            ↪ inclusive.
        if (x > max) {
            uint256 diff = x - max;
            uint256 rem = diff % size;
            if (rem == 0) return max;
            result = min + rem - 1;
        } else if (x < min) {
            uint256 diff = min - x;
            uint256 rem = diff % size;
            if (rem == 0) return min;
            result = max - rem + 1;
        }
    }
```

8.12.21 bound(x, min, max) [StdUtils]

```
    function bound(uint256 x, uint256 min, uint256 max) internal view virtual
        ↪ returns (uint256 result) {
        result = _bound(x, min, max);
        console2_log("Bound Result", result);
    }
```

8.12.22 bound(x, min, max) [StdUtils]

```
    function bound(int256 x, int256 min, int256 max) internal view virtual
        ↪ returns (int256 result) {
        require(min <= max, "StdUtils bound(int256,int256,int256): Max is less
            ↪ than min.");

        // Shifting all int256 values to uint256 to use _bound function. The
            ↪ range of two types are:
        // int256 : -(2**255) ~ (2**255 - 1)
        // uint256:     0     ~ (2**256 - 1)
        // So, add 2**255, INT256_MIN_ABS to the integer values.
        //
        // If the given integer value is -2**255, we cannot use `-uint256(-x)`
            ↪ because of the overflow.
        // So, use `~uint256(x) + 1` instead.
        uint256 _x = x < 0 ? (INT256_MIN_ABS - ~uint256(x) - 1) : (uint256(x) +
            ↪ INT256_MIN_ABS);
        uint256 _min = min < 0 ? (INT256_MIN_ABS - ~uint256(min) - 1) : (uint256
            ↪ (min) + INT256_MIN_ABS);
        uint256 _max = max < 0 ? (INT256_MIN_ABS - ~uint256(max) - 1) : (uint256
            ↪ (max) + INT256_MIN_ABS);

        uint256 y = _bound(_x, _min, _max);

        // To move it back to int256 value, subtract INT256_MIN_ABS at here.
        result = y < INT256_MIN_ABS ? int256(~(INT256_MIN_ABS - y) + 1) : int256
            ↪ (y - INT256_MIN_ABS);
        console2_log("Bound result", vm.toString(result));
    }
```

8.12.23 computeCreateAddress(deployer, nonce) [StdUtils]

```
    /// @dev Compute the address a contract will be deployed at for a given
        ↪ deployer address and nonce
    /// @notice adapated from Solmate implementation (https://github.com/Rari-
        ↪ Capital/solmate/blob/main/src/utils/LibRLP.sol)
    function computeCreateAddress(address deployer, uint256 nonce) internal pure
        ↪  virtual returns (address) {
        // forgefmt: disable-start
        // The integer zero is treated as an empty byte string, and as a result
            ↪ it only has a length prefix, 0x80, computed via 0x80 + 0.
        // A one byte integer uses its own value as its length prefix, there is
            ↪ no additional "0x80 + length" prefix that comes before it.
        if (nonce == 0x00)      return addressFromLast20Bytes(keccak256(abi.
            ↪ encodePacked(bytes1(0xd6), bytes1(0x94), deployer, bytes1(0x80))
            ↪ );
        if (nonce <= 0x7f)      return addressFromLast20Bytes(keccak256(abi.
            ↪ encodePacked(bytes1(0xd6), bytes1(0x94), deployer, uint8(nonce))
            ↪ );

        // Nonces greater than 1 byte all follow a consistent encoding scheme,
            ↪ where each value is preceded by a prefix of 0x80 + length.
        if (nonce <= 2**8 - 1)  return addressFromLast20Bytes(keccak256(abi.
            ↪ encodePacked(bytes1(0xd7), bytes1(0x94), deployer, bytes1(0x81),
            ↪ uint8(nonce))));
        if (nonce <= 2**16 - 1) return addressFromLast20Bytes(keccak256(abi.
            ↪ encodePacked(bytes1(0xd8), bytes1(0x94), deployer, bytes1(0x82),
            ↪ uint16(nonce))));
        if (nonce <= 2**24 - 1) return addressFromLast20Bytes(keccak256(abi.
            ↪ encodePacked(bytes1(0xd9), bytes1(0x94), deployer, bytes1(0x83),
            ↪ uint24(nonce))));
        // forgefmt: disable-end

        // More details about RLP encoding can be found here: https://eth.wiki/
            ↪ fundamentals/rlp
        // 0xda = 0xc0 (short RLP prefix) + 0x16 (length of: 0x94 ++ proxy ++ 0
            ↪ x84 ++ nonce)
        // 0x94 = 0x80 + 0x14 (0x14 = the length of an address, 20 bytes, in hex
            ↪ )
        // 0x84 = 0x80 + 0x04 (0x04 = the bytes length of the nonce, 4 bytes, in
            ↪  hex)
        // We assume nobody can have a nonce large enough to require more than
            ↪ 32 bytes.
        return addressFromLast20Bytes(
            keccak256(abi.encodePacked(bytes1(0xda), bytes1(0x94), deployer,
                ↪ bytes1(0x84), uint32(nonce)))
        );
    }
```

8.12.24 computeCreate2Address(salt, initcodeHash, deployer) [StdUtils]

```
    function computeCreate2Address(bytes32 salt, bytes32 initcodeHash, address
        ↪ deployer)
        internal
        pure
        virtual
        returns (address)
    {
        return addressFromLast20Bytes(keccak256(abi.encodePacked(bytes1(0xff),
            ↪ deployer, salt, initcodeHash)));
    }
```

8.12.25 bytesToUint(b) [StdUtils]

```
    function bytesToUint(bytes memory b) internal pure virtual returns (uint256)
        ↪  {
        require(b.length <= 32, "StdUtils bytesToUint(bytes): Bytes length
            ↪ exceeds 32.");
        return abi.decode(abi.encodePacked(new bytes(32 - b.length), b), (
            ↪ uint256));
    }
```

8.12.26 addressFromLast20Bytes(bytesValue) [StdUtils]

```
function addressFromLast20Bytes(bytes32 bytesValue) private pure returns (
    ↪ address) {
    return address(uint160(uint256(bytesValue)));
}
```

8.12.27 console2_log(p0, p1) [StdUtils]

```
// Used to prevent the compilation of console, which shortens the
    ↪ compilation time when console is not used elsewhere.

function console2_log(string memory p0, uint256 p1) private view {
    (bool status,) = address(CONSOLE2_ADDRESS).staticcall(abi.
        ↪ encodeWithSignature("log(string,uint256)", p0, p1));
    status;
}
```

8.12.28 console2_log(p0, p1) [StdUtils]

```
function console2_log(string memory p0, string memory p1) private view {
    (bool status,) = address(CONSOLE2_ADDRESS).staticcall(abi.
        ↪ encodeWithSignature("log(string,string)", p0, p1));
    status;
}
```

8.12.29 assumeNoPrecompiles(addr) [StdCheatsSafe]

```
function assumeNoPrecompiles(address addr) internal virtual {
    // Assembly required since `block.chainid` was introduced in 0.8.0.
    uint256 chainId;
    assembly {
        chainId := chainid()
    }
    assumeNoPrecompiles(addr, chainId);
}
```

8.12.30 assumeNoPrecompiles(addr, chainId) [StdCheatsSafe]

```
function assumeNoPrecompiles(address addr, uint256 chainId) internal pure
    ↪ virtual {
    // Note: For some chains like Optimism these are technically predeploys
        ↪ (i.e. bytecode placed at a specific
    // address), but the same rationale for excluding them applies so we
        ↪ include those too.

    // These should be present on all EVM-compatible chains.
    vm.assume(addr < address(0x1) || addr > address(0x9));

    // forgefmt: disable-start
    if (chainId == 10 || chainId == 420) {
        // https://github.com/ethereum-optimism/optimism/blob/
            ↪ eaa371a0184b56b7ca6d9eb9cb0a2b78b2ccd864/op-bindings/
            ↪ predeploys/addresses.go#L6-L21
        vm.assume(addr < address(0x4200000000000000000000000000000000000000)
            ↪  || addr > address(0x4200000000000000000000000000000000000800
            ↪ ));
    } else if (chainId == 42161 || chainId == 421613) {
        // https://developer.arbitrum.io/useful-addresses#arbitrum-
            ↪ precompiles-l2-same-on-all-arb-chains
        vm.assume(addr < address(0x0000000000000000000000000000000000000064)
            ↪  || addr > address(0x0000000000000000000000000000000000000068
            ↪ ));
    } else if (chainId == 43114 || chainId == 43113) {
        // https://github.com/ava-labs/subnet-evm/blob/47
            ↪ c03fd007ecaa6de2c52ea081596e0a88401f58/precompile/params.go#
            ↪ L18-L59
        vm.assume(addr < address(0x0100000000000000000000000000000000000000)
            ↪  || addr > address(0x01000000000000000000000000000000000000ff
            ↪ ));
        vm.assume(addr < address(0x0200000000000000000000000000000000000000)
            ↪  || addr > address(0x02000000000000000000000000000000000000FF
            ↪ ));
        vm.assume(addr < address(0x0300000000000000000000000000000000000000)
            ↪  || addr > address(0x03000000000000000000000000000000000000Ff
            ↪ ));
    }
    // forgefmt: disable-end
}
```

8.12.31 readEIP1559ScriptArtifact(path) [StdCheatsSafe]

```
    function readEIP1559ScriptArtifact(string memory path)
        internal
        view
        virtual
        returns (EIP1559ScriptArtifact memory)
    {
        string memory data = vm.readFile(path);
        bytes memory parsedData = vm.parseJson(data);
        RawEIP1559ScriptArtifact memory rawArtifact = abi.decode(parsedData, (
            ↪ RawEIP1559ScriptArtifact));
        EIP1559ScriptArtifact memory artifact;
        artifact.libraries = rawArtifact.libraries;
        artifact.path = rawArtifact.path;
        artifact.timestamp = rawArtifact.timestamp;
        artifact.pending = rawArtifact.pending;
        artifact.txReturns = rawArtifact.txReturns;
        artifact.receipts = rawToConvertedReceipts(rawArtifact.receipts);
        artifact.transactions = rawToConvertedEIPTx1559s(rawArtifact.
            ↪ transactions);
        return artifact;
    }
```

8.12.32 rawToConvertedEIPTx1559s(rawTxs) [StdCheatsSafe]

```
    function rawToConvertedEIPTx1559s(RawTx1559[] memory rawTxs) internal pure
       ↪ virtual returns (Tx1559[] memory) {
        Tx1559[] memory txs = new Tx1559[](rawTxs.length);
        for (uint256 i; i < rawTxs.length; i++) {
            txs[i] = rawToConvertedEIPTx1559(rawTxs[i]);
        }
        return txs;
    }
```

8.12.33 rawToConvertedEIPTx1559(rawTx) [StdCheatsSafe]

```
    function rawToConvertedEIPTx1559(RawTx1559 memory rawTx) internal pure
       ↪ virtual returns (Tx1559 memory) {
        Tx1559 memory transaction;
        transaction.arguments = rawTx.arguments;
        transaction.contractName = rawTx.contractName;
        transaction.functionSig = rawTx.functionSig;
        transaction.hash = rawTx.hash;
        transaction.txDetail = rawToConvertedEIP1559Detail(rawTx.txDetail);
        transaction.opcode = rawTx.opcode;
        return transaction;
    }
```

8.12.34 rawToConvertedEIP1559Detail(rawDetail) [StdCheatsSafe]

```
    function rawToConvertedEIP1559Detail(RawTx1559Detail memory rawDetail)
        internal
        pure
        virtual
        returns (Tx1559Detail memory)
    {
        Tx1559Detail memory txDetail;
        txDetail.data = rawDetail.data;
        txDetail.from = rawDetail.from;
        txDetail.to = rawDetail.to;
        txDetail.nonce = _bytesToUint(rawDetail.nonce);
        txDetail.txType = _bytesToUint(rawDetail.txType);
        txDetail.value = _bytesToUint(rawDetail.value);
        txDetail.gas = _bytesToUint(rawDetail.gas);
        txDetail.accessList = rawDetail.accessList;
        return txDetail;
    }
```

8.12.35 readTx1559s(path) [StdCheatsSafe]

```
    function readTx1559s(string memory path) internal view virtual returns (
       ↪ Tx1559[] memory) {
        string memory deployData = vm.readFile(path);
        bytes memory parsedDeployData = vm.parseJson(deployData, ".transactions"
           ↪ );
        RawTx1559[] memory rawTxs = abi.decode(parsedDeployData, (RawTx1559[]));
        return rawToConvertedEIPTx1559s(rawTxs);
    }
```

8.12.36 readTx1559(path, index) [StdCheatsSafe]

```
    function readTx1559(string memory path, uint256 index) internal view virtual
       ↪  returns (Tx1559 memory) {
        string memory deployData = vm.readFile(path);
        string memory key = string(abi.encodePacked(".transactions[", vm.
           ↪ toString(index), "]"));
        bytes memory parsedDeployData = vm.parseJson(deployData, key);
        RawTx1559 memory rawTx = abi.decode(parsedDeployData, (RawTx1559));
        return rawToConvertedEIPTx1559(rawTx);
    }
```

8.12.37 readReceipts(path) [StdCheatsSafe]

```
    // Analogous to readTransactions, but for receipts.
    function readReceipts(string memory path) internal view virtual returns (
       ↪ Receipt[] memory) {
        string memory deployData = vm.readFile(path);
        bytes memory parsedDeployData = vm.parseJson(deployData, ".receipts");
        RawReceipt[] memory rawReceipts = abi.decode(parsedDeployData, (
           ↪ RawReceipt[]));
        return rawToConvertedReceipts(rawReceipts);
    }
```

8.12.38 readReceipt(path, index) [StdCheatsSafe]

```
    function readReceipt(string memory path, uint256 index) internal view
       ↪ virtual returns (Receipt memory) {
        string memory deployData = vm.readFile(path);
        string memory key = string(abi.encodePacked(".receipts[", vm.toString(
           ↪ index), "]"));
        bytes memory parsedDeployData = vm.parseJson(deployData, key);
        RawReceipt memory rawReceipt = abi.decode(parsedDeployData, (RawReceipt)
            ↪ );
        return rawToConvertedReceipt(rawReceipt);
    }
```

8.12.39 rawToConvertedReceipts(rawReceipts) [StdCheatsSafe]

```
    function rawToConvertedReceipts(RawReceipt[] memory rawReceipts) internal
        ↪ pure virtual returns (Receipt[] memory) {
        Receipt[] memory receipts = new Receipt[](rawReceipts.length);
        for (uint256 i; i < rawReceipts.length; i++) {
            receipts[i] = rawToConvertedReceipt(rawReceipts[i]);
        }
        return receipts;
    }
```

8.12.40 rawToConvertedReceipt(rawReceipt) [StdCheatsSafe]

```
    function rawToConvertedReceipt(RawReceipt memory rawReceipt) internal pure
        ↪ virtual returns (Receipt memory) {
        Receipt memory receipt;
        receipt.blockHash = rawReceipt.blockHash;
        receipt.to = rawReceipt.to;
        receipt.from = rawReceipt.from;
        receipt.contractAddress = rawReceipt.contractAddress;
        receipt.effectiveGasPrice = _bytesToUint(rawReceipt.effectiveGasPrice);
        receipt.cumulativeGasUsed = _bytesToUint(rawReceipt.cumulativeGasUsed);
        receipt.gasUsed = _bytesToUint(rawReceipt.gasUsed);
        receipt.status = _bytesToUint(rawReceipt.status);
        receipt.transactionIndex = _bytesToUint(rawReceipt.transactionIndex);
        receipt.blockNumber = _bytesToUint(rawReceipt.blockNumber);
        receipt.logs = rawToConvertedReceiptLogs(rawReceipt.logs);
        receipt.logsBloom = rawReceipt.logsBloom;
        receipt.transactionHash = rawReceipt.transactionHash;
        return receipt;
    }
```

8.12.41 rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe]

```
    function rawToConvertedReceiptLogs(RawReceiptLog[] memory rawLogs)
        internal
        pure
        virtual
        returns (ReceiptLog[] memory)
    {
        ReceiptLog[] memory logs = new ReceiptLog[](rawLogs.length);
        for (uint256 i; i < rawLogs.length; i++) {
            logs[i].logAddress = rawLogs[i].logAddress;
            logs[i].blockHash = rawLogs[i].blockHash;
            logs[i].blockNumber = _bytesToUint(rawLogs[i].blockNumber);
            logs[i].data = rawLogs[i].data;
            logs[i].logIndex = _bytesToUint(rawLogs[i].logIndex);
            logs[i].topics = rawLogs[i].topics;
            logs[i].transactionIndex = _bytesToUint(rawLogs[i].transactionIndex)
                ↪ ;
            logs[i].transactionLogIndex = _bytesToUint(rawLogs[i].
                ↪ transactionLogIndex);
            logs[i].removed = rawLogs[i].removed;
        }
        return logs;
    }
```

8.12.42 deployCode(what, args) [StdCheatsSafe]

```
    // Deploy a contract by fetching the contract bytecode from
    // the artifacts directory
    // e.g. `deployCode(code, abi.encode(arg1,arg2,arg3))`
    function deployCode(string memory what, bytes memory args) internal virtual
        ↪ returns (address addr) {
        bytes memory bytecode = abi.encodePacked(vm.getCode(what), args);
        /// @solidity memory-safe-assembly
        assembly {
            addr := create(0, add(bytecode, 0x20), mload(bytecode))
        }

        require(addr != address(0), "StdCheats deployCode(string,bytes):
            ↪ Deployment failed.");
    }
```

8.12.43 deployCode(what) [StdCheatsSafe]

```
    function deployCode(string memory what) internal virtual returns (address
        ↪ addr) {
        bytes memory bytecode = vm.getCode(what);
        /// @solidity memory-safe-assembly
        assembly {
            addr := create(0, add(bytecode, 0x20), mload(bytecode))
        }

        require(addr != address(0), "StdCheats deployCode(string): Deployment
            ↪ failed.");
    }
```

8.12.44 deployCode(what, args, val) [StdCheatsSafe]

```
    /// @dev deploy contract with value on construction
    function deployCode(string memory what, bytes memory args, uint256 val)
        ↪ internal virtual returns (address addr) {
        bytes memory bytecode = abi.encodePacked(vm.getCode(what), args);
        /// @solidity memory-safe-assembly
        assembly {
            addr := create(val, add(bytecode, 0x20), mload(bytecode))
        }

        require(addr != address(0), "StdCheats deployCode(string,bytes,uint256):
            ↪  Deployment failed.");
    }
```

8.12.45 deployCode(what, val) [StdCheatsSafe]

```
    function deployCode(string memory what, uint256 val) internal virtual
        ↪ returns (address addr) {
        bytes memory bytecode = vm.getCode(what);
        /// @solidity memory-safe-assembly
        assembly {
            addr := create(val, add(bytecode, 0x20), mload(bytecode))
        }

        require(addr != address(0), "StdCheats deployCode(string,uint256):
            ↪ Deployment failed.");
    }
```

8.12.46 makeAddrAndKey(name) [StdCheatsSafe]

```
    // creates a labeled address and the corresponding private key
    function makeAddrAndKey(string memory name) internal virtual returns (
        ↪ address addr, uint256 privateKey) {
        privateKey = uint256(keccak256(abi.encodePacked(name)));
        addr = vm.addr(privateKey);
        vm.label(addr, name);
    }
```

8.12.47 makeAddr(name) [StdCheatsSafe]

```
    // creates a labeled address
    function makeAddr(string memory name) internal virtual returns (address addr
        ↪ ) {
        (addr,) = makeAddrAndKey(name);
    }
```

8.12.48 deriveRememberKey(mnemonic, index) [StdCheatsSafe]

```
    function deriveRememberKey(string memory mnemonic, uint32 index)
        internal
        virtual
        returns (address who, uint256 privateKey)
    {
        privateKey = vm.deriveKey(mnemonic, index);
        who = vm.rememberKey(privateKey);
    }
```

8.12.49 _bytesToUint(b) [StdCheatsSafe]

```
    function _bytesToUint(bytes memory b) private pure returns (uint256) {
        require(b.length <= 32, "StdCheats _bytesToUint(bytes): Bytes length
            ↪ exceeds 32.");
        return abi.decode(abi.encodePacked(new bytes(32 - b.length), b), (
            ↪ uint256));
    }
```

8.12.50 isFork() [StdCheatsSafe]

```
    function isFork() internal view virtual returns (bool status) {
        try vm.activeFork() {
            status = true;
        } catch (bytes memory) {}
    }
```

8.12.51 getChain(chainAlias) [StdChains]

```
    // The RPC URL will be fetched from config or defaultRpcUrls if possible.
    function getChain(string memory chainAlias) internal virtual returns (Chain
        ↪ memory chain) {
        require(bytes(chainAlias).length != 0, "StdChains getChain(string):
            ↪ Chain alias cannot be the empty string.");

        initialize();
        chain = chains[chainAlias];
        require(
            chain.chainId != 0,
            string(abi.encodePacked("StdChains getChain(string): Chain with
                ↪ alias \"", chainAlias, "\" not found."))
        );
        chain = getChainWithUpdatedRpcUrl(chainAlias, chain);
    }
```

8.12.52 getChain(chainId) [StdChains]

```
    function getChain(uint256 chainId) internal virtual returns (Chain memory
        ↪ chain) {
        require(chainId != 0, "StdChains getChain(uint256): Chain ID cannot be
            ↪ 0.");
        initialize();
        string memory chainAlias = idToAlias[chainId];

        chain = chains[chainAlias];

        require(
            chain.chainId != 0,
            string(abi.encodePacked("StdChains getChain(uint256): Chain with ID
                ↪ ", vm.toString(chainId), " not found."))
        );

        chain = getChainWithUpdatedRpcUrl(chainAlias, chain);
    }
```

8.12.53 setChain(chainAlias, chain) [StdChains]

```
    // set chain info, with priority to argument's rpcUrl field.
    function setChain(string memory chainAlias, ChainData memory chain) internal
        ↪  virtual {
        require(
            bytes(chainAlias).length != 0,
            "StdChains setChain(string,ChainData): Chain alias cannot be the
                ↪ empty string."
        );

        require(chain.chainId != 0, "StdChains setChain(string,ChainData): Chain
            ↪  ID cannot be 0.");

        initialize();
        string memory foundAlias = idToAlias[chain.chainId];

        require(
            bytes(foundAlias).length == 0 || keccak256(bytes(foundAlias)) ==
                ↪ keccak256(bytes(chainAlias)),
            string(
                abi.encodePacked(
                    "StdChains setChain(string,ChainData): Chain ID ",
                    vm.toString(chain.chainId),
                    " already used by \"",
                    foundAlias,
                    "\"."
                )
            )
        );

        uint256 oldChainId = chains[chainAlias].chainId;
        delete idToAlias[oldChainId];

        chains[chainAlias] =
            Chain({name: chain.name, chainId: chain.chainId, chainAlias:
                ↪ chainAlias, rpcUrl: chain.rpcUrl});
        idToAlias[chain.chainId] = chainAlias;
    }
```

8.12.54 setChain(chainAlias, chain) [StdChains]

```
    // set chain info, with priority to argument's rpcUrl field.
    function setChain(string memory chainAlias, Chain memory chain) internal
        ↪ virtual {
        setChain(chainAlias, ChainData({name: chain.name, chainId: chain.chainId
            ↪ , rpcUrl: chain.rpcUrl}));
    }
```

8.12.55 _toUpper(str) [StdChains]

```
    function _toUpper(string memory str) private pure returns (string memory) {
        bytes memory strb = bytes(str);
        bytes memory copy = new bytes(strb.length);
        for (uint256 i = 0; i < strb.length; i++) {
            bytes1 b = strb[i];
            if (b >= 0x61 && b <= 0x7A) {
                copy[i] = bytes1(uint8(b) - 32);
            } else {
                copy[i] = b;
            }
        }
        return string(copy);
    }
```

8.12.56 getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains]

```
    // lookup rpcUrl, in descending order of priority:
    // current -> config (foundry.toml) -> environment variable -> default
    function getChainWithUpdatedRpcUrl(string memory chainAlias, Chain memory
        ↪ chain) private returns (Chain memory) {
        if (bytes(chain.rpcUrl).length == 0) {
            try vm.rpcUrl(chainAlias) returns (string memory configRpcUrl) {
                chain.rpcUrl = configRpcUrl;
            } catch (bytes memory err) {
                chain.rpcUrl =
                    vm.envOr(string(abi.encodePacked(_toUpper(chainAlias), "
                        ↪ _RPC_URL")), defaultRpcUrls[chainAlias]);
                // distinguish 'not found' from 'cannot read'
                bytes memory notFoundError =
                    abi.encodeWithSignature("CheatCodeError", string(abi.
                        ↪ encodePacked("invalid rpc url ", chainAlias)));
                if (keccak256(notFoundError) != keccak256(err) || bytes(chain.
                    ↪ rpcUrl).length == 0) {
                    /// @solidity memory-safe-assembly
                    assembly {
                        revert(add(32, err), mload(err))
                    }
                }
            }
        }
        return chain;
    }
```

8.12.57 initialize() [StdChains]

```
    function initialize() private {
        if (initialized) return;

        initialized = true;

        // If adding an RPC here, make sure to test the default RPC URL in `
            ↪ testRpcs`
        setChainWithDefaultRpcUrl("anvil", ChainData("Anvil", 31337, "http
            ↪ ://127.0.0.1:8545"));
        setChainWithDefaultRpcUrl(
            "mainnet", ChainData("Mainnet", 1, "https://mainnet.infura.io/v3
                ↪ /6770454bc6ea42c58aac12978531b93f")
        );
        setChainWithDefaultRpcUrl(
            "goerli", ChainData("Goerli", 5, "https://goerli.infura.io/v3
                ↪ /6770454bc6ea42c58aac12978531b93f")
        );
        setChainWithDefaultRpcUrl(
            "sepolia", ChainData("Sepolia", 11155111, "https://sepolia.infura.io
                ↪ /v3/6770454bc6ea42c58aac12978531b93f")
        );
        setChainWithDefaultRpcUrl("optimism", ChainData("Optimism", 10, "https
            ↪ ://mainnet.optimism.io"));
        setChainWithDefaultRpcUrl("optimism_goerli", ChainData("Optimism Goerli"
            ↪ , 420, "https://goerli.optimism.io"));
        setChainWithDefaultRpcUrl("arbitrum_one", ChainData("Arbitrum One",
            ↪ 42161, "https://arb1.arbitrum.io/rpc"));
        setChainWithDefaultRpcUrl(
            "arbitrum_one_goerli", ChainData("Arbitrum One Goerli", 421613, "
                ↪ https://goerli-rollup.arbitrum.io/rpc")
        );
        setChainWithDefaultRpcUrl("arbitrum_nova", ChainData("Arbitrum Nova",
            ↪ 42170, "https://nova.arbitrum.io/rpc"));
        setChainWithDefaultRpcUrl("polygon", ChainData("Polygon", 137, "https://
            ↪ polygon-rpc.com"));
        setChainWithDefaultRpcUrl(
            "polygon_mumbai", ChainData("Polygon Mumbai", 80001, "https://rpc-
                ↪ mumbai.maticvigil.com")
        );
        setChainWithDefaultRpcUrl("avalanche", ChainData("Avalanche", 43114, "
            ↪ https://api.avax.network/ext/bc/C/rpc"));
        setChainWithDefaultRpcUrl(
            "avalanche_fuji", ChainData("Avalanche Fuji", 43113, "https://api.
                ↪ avax-test.network/ext/bc/C/rpc")
        );
        setChainWithDefaultRpcUrl(
            "bnb_smart_chain", ChainData("BNB Smart Chain", 56, "https://bsc-
                ↪ dataseed1.binance.org")
        );
        setChainWithDefaultRpcUrl(
            "bnb_smart_chain_testnet",
            ChainData("BNB Smart Chain Testnet", 97, "https://data-seed-prebsc
                ↪ -1-s1.binance.org:8545")
        );
        setChainWithDefaultRpcUrl("gnosis_chain", ChainData("Gnosis Chain", 100,
            ↪  "https://rpc.gnosischain.com"));
    }
```

8.12.58 setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains]

```
    // set chain info, with priority to chainAlias' rpc url in foundry.toml
    function setChainWithDefaultRpcUrl(string memory chainAlias, ChainData
        ↪ memory chain) private {
        string memory rpcUrl = chain.rpcUrl;
        defaultRpcUrls[chainAlias] = rpcUrl;
        chain.rpcUrl = "";
        setChain(chainAlias, chain);
        chain.rpcUrl = rpcUrl; // restore argument
    }
```

8.12.59 run() X

```
    function run() external {
        config = ScriptTools.loadConfig();
        dependencies = ScriptTools.loadDependencies();
        dss = MCD.loadFromChainlog(config.readAddress("chainlog"));

        d3mType = config.readString("type");
        ilk = config.readString("ilk").stringToBytes32();

        d3m = D3MInstance({
            pool: dependencies.readAddress("pool"),
            plan: dependencies.readAddress("plan"),
            oracle: dependencies.readAddress("oracle")
        });
        cfg = D3MCommonConfig({
            hub: dependencies.readAddress("hub"),
            mom: dependencies.readAddress("mom"),
            ilk: ilk,
            existingIlk: config.readBool("existingIlk"),
            maxLine: config.readUint("maxLine") * RAD,
            gap: config.readUint("gap") * RAD,
            ttl: config.readUint("ttl"),
            tau: config.readUint("tau")
        });

        vm.startBroadcast();
        if (d3mType.eq("aave")) {
            aaveCfg = D3MAaveConfig({
                king: config.readAddress("king"),
                bar: config.readUint("bar") * RAY / BPS,
                adai: AavePoolLike(d3m.pool).adai(),
                stableDebt: AavePoolLike(d3m.pool).stableDebt(),
                variableDebt: AavePoolLike(d3m.pool).variableDebt(),
                tack: AavePlanLike(d3m.plan).tack(),
                adaiRevision: AavePlanLike(d3m.plan).adaiRevision()
            });
            D3MInit.initAave(
                dss,
                d3m,
                cfg,
                aaveCfg
            );
        } else if (d3mType.eq("compound")) {
            compoundCfg = D3MCompoundConfig({
                king: config.readAddress("king"),
                barb: config.readUint("barb"),
                cdai: CompoundPoolLike(d3m.pool).cDai(),
                comptroller: CompoundPoolLike(d3m.pool).comptroller(),
                comp: CompoundPoolLike(d3m.pool).comp(),
                tack: CompoundPlanLike(d3m.plan).tack(),
                delegate: CompoundPlanLike(d3m.plan).delegate()
            });
            D3MInit.initCompound(
                dss,
                d3m,
                cfg,
                compoundCfg
            );
        } else {
            revert("unknown-d3m-type");
        }
        vm.stopBroadcast();
    }
```

Chapter 9

Gem joins

9.1 contract GemJoin

```
/*
    Here we provide *adapters* to connect the Vat to arbitrary external
    token implementations, creating a bounded context for the Vat. The
    adapters here are provided as working examples:

      - `GemJoin`: For well behaved ERC20 tokens, with simple transfer
                   semantics.

      - `ETHJoin`: For native Ether.

      - `DaiJoin`: For connecting internal Dai balances to an external
                   `DSToken` implementation.

    In practice, adapter implementations will be varied and specific to
    individual collateral types, accounting for different transfer
    semantics and token standards.

    Adapters need to implement two basic methods:

      - `join`: enter collateral into the system
      - `exit`: remove collateral from the system

*/

contract GemJoin {
    // --- Auth ---
    mapping (address => uint) public wards;

    VatLike public vat;   // CDP Engine
    bytes32 public ilk;   // Collateral Type
    GemLike public gem;
    uint    public dec;
    uint    public live;  // Active Flag

    // Events
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event Join(address indexed usr, uint256 wad);
    event Exit(address indexed usr, uint256 wad);
    event Cage();
}
```

9.1.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "GemJoin/not-authorized");
        _;
    }
```

9.1.2 rely(usr) X a

```
function rely(address usr) external auth {
    wards[usr] = 1;
    emit Rely(usr);
}
```

9.1.3 deny(usr) X a

```
function deny(address usr) external auth {
    wards[usr] = 0;
    emit Deny(usr);
}
```

9.1.4 constructor(vat_, ilk_, gem_) X

```
constructor(address vat_, bytes32 ilk_, address gem_) public {
    wards[msg.sender] = 1;
    live = 1;
    vat = VatLike(vat_);
    ilk = ilk_;
    gem = GemLike(gem_);
    dec = gem.decimals();
    emit Rely(msg.sender);
}
```

9.1.5 cage() X a

```
function cage() external auth {
    live = 0;
    emit Cage();
}
```

9.1.6 join(usr, wad) X

```
function join(address usr, uint wad) external {
    require(live == 1, "GemJoin/not-live");
    require(int(wad) >= 0, "GemJoin/overflow");
    vat.slip(ilk, usr, int(wad));
    require(gem.transferFrom(msg.sender, address(this), wad), "GemJoin/
        ↪ failed-transfer");
    emit Join(usr, wad);
}
```

9.1.7 exit(usr, wad) X

```
function exit(address usr, uint wad) external {
    require(wad <= 2 ** 255, "GemJoin/overflow");
    vat.slip(ilk, msg.sender, -int(wad));
    require(gem.transfer(usr, wad), "GemJoin/failed-transfer");
    emit Exit(usr, wad);
}
```

9.2 contract DaiJoin

```
contract DaiJoin {
    // --- Auth ---
    mapping (address => uint) public wards;

    VatLike public vat;       // CDP Engine
    DSTokenLike public dai;   // Stablecoin Token
    uint    public live;      // Active Flag

    // Events
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event Join(address indexed usr, uint256 wad);
    event Exit(address indexed usr, uint256 wad);
    event Cage();
    uint constant ONE = 10 ** 27;
}
```

9.2.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "DaiJoin/not-authorized");
        _;
    }
```

9.2.2 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

9.2.3 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

9.2.4 constructor(vat_, dai_) X

```
    constructor(address vat_, address dai_) public {
        wards[msg.sender] = 1;
        live = 1;
        vat = VatLike(vat_);
        dai = DSTokenLike(dai_);
    }
```

9.2.5 cage() X a

```
    function cage() external auth {
        live = 0;
        emit Cage();
    }
```

9.2.6 mul(x, y)

```
    function mul(uint x, uint y) internal pure returns (uint z) {
        require(y == 0 || (z = x * y) / y == x);
    }
```

9.2.7 join(usr, wad) X

```
    function join(address usr, uint wad) external {
        vat.move(address(this), usr, mul(ONE, wad));
        dai.burn(msg.sender, wad);
        emit Join(usr, wad);
    }
```

9.2.8 exit(usr, wad) X

```
    function exit(address usr, uint wad) external {
        require(live == 1, "DaiJoin/not-live");
        vat.move(msg.sender, address(this), mul(ONE, wad));
        dai.mint(usr, wad);
        emit Exit(usr, wad);
    }
```

9.3 contract GemJoin2

```
// For a token that does not return a bool on transfer or transferFrom (like OMG
    ↪ )
// This is one way of doing it. Check the balances before and after calling a
    ↪ transfer

contract GemJoin2 {
    // --- Auth ---
    mapping (address => uint256) public wards;

    VatLike public vat;
    bytes32 public ilk;
    GemLike public gem;
    uint256 public dec;
    uint256 public live;  // Access Flag

    // Events
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event Join(address indexed usr, uint256 wad);
    event Exit(address indexed usr, uint256 wad);
    event Cage();
}
```

9.3.1 modifier auth()

```
    modifier auth { require(wards[msg.sender] == 1); _; }
```

9.3.2 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

9.3.3 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

9.3.4 constructor(vat_, ilk_, gem_) X

```
    constructor(address vat_, bytes32 ilk_, address gem_) public {
        wards[msg.sender] = 1;
        live = 1;
        vat = VatLike(vat_);
        ilk = ilk_;
        gem = GemLike(gem_);
        dec = gem.decimals();
        emit Rely(msg.sender);
    }
```

9.3.5 cage() X a

```
    function cage() external auth {
        live = 0;
        emit Cage();
    }
```

9.3.6 mul(x, y)

```
    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require(y == 0 || (z = x * y) / y == x, "GemJoin2/overflow");
    }
```

9.3.7 join(usr, wad) X

```
    function join(address usr, uint256 wad) external {
        require(live == 1, "GemJoin2/not-live");
        require(wad <= 2 ** 255, "GemJoin2/overflow");
        vat.slip(ilk, usr, int256(wad));
        uint256 prevBalance = gem.balanceOf(msg.sender);

        require(prevBalance >= wad, "GemJoin2/no-funds");
        require(gem.allowance(msg.sender, address(this)) >= wad, "GemJoin2/no-
            ↪ allowance");

        (bool ok,) = address(gem).call(
            abi.encodeWithSignature("transferFrom(address,address,uint256)", msg
                ↪ .sender, address(this), wad)
        );
        require(ok, "GemJoin2/failed-transfer");

        require(prevBalance - wad == gem.balanceOf(msg.sender), "GemJoin2/failed
            ↪ -transfer");

        emit Join(usr, wad);
    }
```

9.3.8 exit(usr, wad) X

```
    function exit(address usr, uint256 wad) external {
        require(wad <= 2 ** 255, "GemJoin2/overflow");
        vat.slip(ilk, msg.sender, -int256(wad));
        uint256 prevBalance = gem.balanceOf(address(this));

        require(prevBalance >= wad, "GemJoin2/no-funds");

        (bool ok,) = address(gem).call(
            abi.encodeWithSignature("transfer(address,uint256)", usr, wad)
        );
        require(ok, "GemJoin2/failed-transfer");

        require(prevBalance - wad == gem.balanceOf(address(this)), "GemJoin2/
            ↪ failed-transfer");

        emit Exit(usr, wad);
    }
```

9.4 contract GemJoin3

```
// For a token that has a lower precision than 18 and doesn't have decimals
    ↪ field in place (like DGD)

contract GemJoin3 {
    // --- Auth ---
    mapping (address => uint256) public wards;

    VatLike public vat;
    bytes32 public ilk;
    GemLike public gem;
    uint256 public dec;
    uint256 public live;  // Access Flag

    // Events
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event Join(address indexed usr, uint256 wad);
    event Exit(address indexed usr, uint256 wad);
    event Cage();
}
```

9.4.1 modifier auth()

```
    modifier auth { require(wards[msg.sender] == 1); _; }
```

9.4.2 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

9.4.3 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

9.4.4 constructor(vat_, ilk_, gem_, decimals) X

```
    constructor(address vat_, bytes32 ilk_, address gem_, uint256 decimals)
        ↪ public {
        require(decimals < 18, "GemJoin3/decimals-18-or-higher");
        wards[msg.sender] = 1;
        live = 1;
        vat = VatLike(vat_);
        ilk = ilk_;
        gem = GemLike(gem_);
        dec = decimals;
        emit Rely(msg.sender);
    }
```

9.4.5 cage() X a

```
    function cage() external auth {
        live = 0;
        emit Cage();
    }
```

9.4.6 mul(x, y)

```
    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require(y == 0 || (z = x * y) / y == x, "GemJoin3/overflow");
    }
```

9.4.7 join(usr, amt) X

```
    function join(address usr, uint256 amt) external {
        require(live == 1, "GemJoin3/not-live");
        uint256 wad = mul(amt, 10 ** (18 - dec));
        require(wad <= 2 ** 255, "GemJoin3/overflow");
        vat.slip(ilk, usr, int256(wad));
        require(gem.transferFrom(msg.sender, address(this), amt), "GemJoin3/
           ↪ failed-transfer");
        emit Join(usr, amt);
    }
```

9.4.8 exit(usr, amt) X

```
    function exit(address usr, uint256 amt) external {
        uint256 wad = mul(amt, 10 ** (18 - dec));
        require(wad <= 2 ** 255, "GemJoin3/overflow");
        vat.slip(ilk, msg.sender, -int256(wad));
        require(gem.transfer(usr, amt), "GemJoin3/failed-transfer");
        emit Exit(usr, amt);
    }
```

9.5 contract GemBag

```
// For tokens that do not implement transferFrom (like GNT), meaning the usual
   ↪ adapter
// approach won't work: the adapter cannot call transferFrom and therefore
// has no way of knowing when users deposit gems into it.

// To work around this, we introduce the concept of a bag, which is a trusted
// (it's created by the adapter), personalized component (one for each user).

// Users first have to create their bag with `GemJoin4.make`, then transfer
// gem to it, and then call `GemJoin4.join`, which transfer the gems from the
// bag to the adapter.

contract GemBag {
    address public ada;
    address public lad;
    GemLike public gem;
}
```

9.5.1 constructor(lad_, gem_) X

```
    constructor(address lad_, address gem_) public {
        ada = msg.sender;
        lad = lad_;
        gem = GemLike(gem_);
    }
```

9.5.2 exit(usr, wad) X

```
    function exit(address usr, uint256 wad) external {
        require(msg.sender == ada || msg.sender == lad, "GemBag/invalid-caller")
            ↪ ;
        require(gem.transfer(usr, wad), "GemBag/failed-transfer");
    }
```

9.6 contract GemJoin4

```
contract GemJoin4 {
    // --- Auth ---
    mapping (address => uint256) public wards;

    VatLike public vat;
    bytes32 public ilk;
    GemLike public gem;
    uint256 public dec;
    uint256 public live;  // Access Flag

    // Events
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event Join(address indexed usr, uint256 wad);
    event Exit(address indexed usr, uint256 wad);
    event Cage();

    mapping(address => address) public bags;
}
```

9.6.1 modifier auth()

```
    modifier auth { require(wards[msg.sender] == 1); _; }
```

9.6.2 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

9.6.3 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

9.6.4 constructor(vat_, ilk_, gem_) X

```
    constructor(address vat_, bytes32 ilk_, address gem_) public {
        wards[msg.sender] = 1;
        live = 1;
        vat = VatLike(vat_);
        ilk = ilk_;
        gem = GemLike(gem_);
        dec = gem.decimals();
        emit Rely(msg.sender);
    }
```

9.6.5 cage() X a

```
    function cage() external auth {
        live = 0;
        emit Cage();
    }
```

9.6.6 make() X

```
    // -- admin --
    function make() external returns (address bag) {
        bag = make(msg.sender);
    }
```

9.6.7 make(usr) X

```
    function make(address usr) public returns (address bag) {
        require(bags[usr] == address(0), "GemJoin4/bag-already-exists");

        bag = address(new GemBag(address(usr), address(gem)));
        bags[usr] = bag;
    }
```

9.6.8 join(usr, wad) X

```
    // -- gems --
    function join(address usr, uint256 wad) external {
        require(live == 1, "GemJoin4/not-live");
        require(int256(wad) >= 0, "GemJoin4/negative-amount");

        GemBag(bags[msg.sender]).exit(address(this), wad);
        vat.slip(ilk, usr, int256(wad));
        emit Join(usr, wad);
    }
```

9.6.9 exit(usr, wad) X

```
    function exit(address usr, uint256 wad) external {
        require(int256(wad) >= 0, "GemJoin4/negative-amount");

        vat.slip(ilk, msg.sender, -int256(wad));
        require(gem.transfer(usr, wad), "GemJoin4/failed-transfer");
        emit Exit(usr, wad);
    }
```

9.7 contract GemJoin5

```
// For a token that has a lower precision than 18 and it has decimals (like USDC
    ↪ )

contract GemJoin5 {
    // --- Auth ---
    mapping (address => uint256) public wards;

    VatLike public vat;
    bytes32 public ilk;
    GemLike public gem;
    uint256 public dec;
    uint256 public live;  // Access Flag

    // Events
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event Join(address indexed usr, uint256 wad);
    event Exit(address indexed usr, uint256 wad);
    event Cage();
}
```

9.7.1 modifier auth()

```
    modifier auth { require(wards[msg.sender] == 1); _; }
```

9.7.2 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

9.7.3 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

9.7.4 constructor(vat_, ilk_, gem_) X

```
    constructor(address vat_, bytes32 ilk_, address gem_) public {
        gem = GemLike(gem_);
        dec = gem.decimals();
        require(dec < 18, "GemJoin5/decimals-18-or-higher");
        wards[msg.sender] = 1;
        live = 1;
        vat = VatLike(vat_);
        ilk = ilk_;
        emit Rely(msg.sender);
    }
```

9.7.5 cage() X a

```
    function cage() external auth {
        live = 0;
        emit Cage();
    }
```

9.7.6 mul(x, y)

```
    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require(y == 0 || (z = x * y) / y == x, "GemJoin5/overflow");
    }
```

9.7.7 join(usr, amt) X

```
    function join(address usr, uint256 amt) external {
        require(live == 1, "GemJoin5/not-live");
        uint256 wad = mul(amt, 10 ** (18 - dec));
        require(int256(wad) >= 0, "GemJoin5/overflow");
        vat.slip(ilk, usr, int256(wad));
        require(gem.transferFrom(msg.sender, address(this), amt), "GemJoin5/
            ↪ failed-transfer");
        emit Join(usr, amt);
    }
```

9.7.8 exit(usr, amt) X

```
    function exit(address usr, uint256 amt) external {
        uint256 wad = mul(amt, 10 ** (18 - dec));
        require(int256(wad) >= 0, "GemJoin5/overflow");
        vat.slip(ilk, msg.sender, -int256(wad));
        require(gem.transfer(usr, amt), "GemJoin5/failed-transfer");
        emit Exit(usr, amt);
    }
```

9.8 contract GemJoin6

```
// For a token with a proxy and implementation contract (like tUSD)
//  If the implementation behind the proxy is changed, this prevents joins
//   and exits until the implementation is reviewed and approved by governance.

contract GemJoin6 {
    // --- Auth ---
    mapping (address => uint256) public wards;

    VatLike public vat;
    bytes32 public ilk;
    GemLike public gem;
    uint256 public dec;
    uint256 public live;  // Access Flag

    // Events
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event Join(address indexed usr, uint256 wad);
    event Exit(address indexed usr, uint256 wad);
    event Cage();

    mapping (address => uint256) public implementations;
}
```

9.8.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "GemJoin6/not-authorized");
        _;
    }
```

9.8.2 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

9.8.3 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

9.8.4 constructor(vat_, ilk_, gem_) X

```
    constructor(address vat_, bytes32 ilk_, address gem_) public {
        wards[msg.sender] = 1;
        live = 1;
        vat = VatLike(vat_);
        ilk = ilk_;
        gem = GemLike(gem_);
        setImplementation(gem.implementation(), 1);
        dec = gem.decimals();
        emit Rely(msg.sender);
    }
```

9.8.5 cage() X a

```
    function cage() external auth {
        live = 0;
        emit Cage();
    }
```

9.8.6 setImplementation(implementation, permitted) X a

```
    function setImplementation(address implementation, uint256 permitted) public
        ↪ auth {
        implementations[implementation] = permitted;  // 1 live, 0 disable
    }
```

9.8.7 join(usr, wad) X

```
    function join(address usr, uint256 wad) external {
        require(live == 1, "GemJoin6/not-live");
        require(int256(wad) >= 0, "GemJoin6/overflow");
        require(implementations[gem.implementation()] == 1, "GemJoin6/
            ↪ implementation-invalid");
        vat.slip(ilk, usr, int256(wad));
        require(gem.transferFrom(msg.sender, address(this), wad), "GemJoin6/
            ↪ failed-transfer");
        emit Join(usr, wad);
    }
```

9.8.8 exit(usr, wad) X

```
    function exit(address usr, uint256 wad) external {
        require(wad <= 2 ** 255, "GemJoin6/overflow");
        require(implementations[gem.implementation()] == 1, "GemJoin6/
            ↪ implementation-invalid");
        vat.slip(ilk, msg.sender, -int256(wad));
        require(gem.transfer(usr, wad), "GemJoin6/failed-transfer");
        emit Exit(usr, wad);
    }
```

9.9 contract GemJoin7

```
// GemJoin7
// For an upgradable token (like USDT) which doesn't return bool on transfers
   ↪ and may charge fees
//  If the token is deprecated changing the implementation behind, this prevents
   ↪  joins
//   and exits until the implementation is reviewed and approved by governance.

contract GemJoin7 {
    mapping (address => uint256) public wards;

    VatLike public vat;
    bytes32 public ilk;
    GemLike public gem;
    uint256 public dec;
    uint256 public live; // Access flag

    // Events
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event Join(address indexed usr, uint256 wad);
    event Exit(address indexed usr, uint256 wad);
    event Cage();

    mapping (address => uint256) public implementations;
}
```

9.9.1 modifier auth()

```
    modifier auth { require(wards[msg.sender] == 1); _; }
```

9.9.2 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

9.9.3 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

9.9.4 constructor(vat_, ilk_, gem_) X

```
    constructor(address vat_, bytes32 ilk_, address gem_) public {
        gem = GemLike(gem_);
        dec = gem.decimals();
        require(dec < 18, "GemJoin7/decimals-18-or-higher");
        wards[msg.sender] = 1;
        live = 1;
        vat = VatLike(vat_);
        ilk = ilk_;
        setImplementation(address(gem.upgradedAddress()), 1);
        emit Rely(msg.sender);
    }
```

9.9.5 cage() X a

```
    function cage() external auth {
        live = 0;
        emit Cage();
    }
```

9.9.6 setImplementation(implementation, permitted) X a

```
    function setImplementation(address implementation, uint256 permitted) public
        ↪  auth {
        implementations[implementation] = permitted; // 1 live, 0 disable
    }
```

9.9.7 mul(x, y)

```
    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require(y == 0 || (z = x * y) / y == x, "GemJoin7/overflow");
    }
```

9.9.8 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "GemJoin7/underflow");
    }
```

9.9.9 join(usr, amt) X

```
    function join(address usr, uint256 amt) external {
        require(live == 1, "GemJoin7/not-live");
        require(implementations[gem.upgradedAddress()] == 1, "GemJoin7/
            ↪ implementation-invalid");
        uint256 bal = gem.balanceOf(address(this));
        gem.transferFrom(msg.sender, address(this), amt);
        uint256 wad = mul(sub(gem.balanceOf(address(this)), bal), 10 ** (18 -
            ↪ dec));
        require(int256(wad) >= 0, "GemJoin7/overflow");
        vat.slip(ilk, usr, int256(wad));
        emit Join(usr, amt);
    }
```

9.9.10 exit(usr, amt) X

```
    function exit(address usr, uint256 amt) external {
        uint256 wad = mul(amt, 10 ** (18 - dec));
        require(int256(wad) >= 0, "GemJoin7/overflow");
        require(implementations[gem.upgradedAddress()] == 1, "GemJoin7/
            ↪ implementation-invalid");
        vat.slip(ilk, msg.sender, -int256(wad));
        gem.transfer(usr, amt);
        emit Exit(usr, amt);
    }
```

9.10 contract GemJoin8

```
// GemJoin8
// For a token that has a lower precision than 18, has decimals and it is
    ↪ upgradable (like GUSD)

contract GemJoin8 {
    // --- Auth ---
    mapping (address => uint256) public wards;

    VatLike  public vat;
    bytes32  public ilk;
    GemLike  public gem;
    uint256  public dec;
    uint256  public live;  // Access Flag

    // Events
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event Join(address indexed usr, uint256 wad);
    event Exit(address indexed usr, uint256 wad);
    event Cage();

    mapping (address => uint256) public implementations;
}
```

9.10.1 modifier auth()

```
    modifier auth { require(wards[msg.sender] == 1); _; }
```

9.10.2 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

9.10.3 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

9.10.4 constructor(vat_, ilk_, gem_) X

```
    constructor(address vat_, bytes32 ilk_, address gem_) public {
        gem = GemLike(gem_);
        dec = gem.decimals();
        require(dec < 18, "GemJoin8/decimals-18-or-higher");
        wards[msg.sender] = 1;
        live = 1;
        setImplementation(gem.erc20Impl(), 1);
        vat = VatLike(vat_);
        ilk = ilk_;
        emit Rely(msg.sender);
    }
```

9.10.5 cage() X a

```
    function cage() external auth {
        live = 0;
        emit Cage();
    }
```

9.10.6 setImplementation(implementation, permitted) X a

```
    function setImplementation(address implementation, uint256 permitted) public
       ↪  auth {
        implementations[implementation] = permitted;  // 1 live, 0 disable
    }
```

9.10.7 mul(x, y)

```
    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require(y == 0 || (z = x * y) / y == x, "GemJoin8/overflow");
    }
```

9.10.8 join(usr, amt) X

```
    function join(address usr, uint256 amt) external {
        require(live == 1, "GemJoin8/not-live");
        uint256 wad = mul(amt, 10 ** (18 - dec));
        require(int256(wad) >= 0, "GemJoin8/overflow");
        require(implementations[gem.erc20Impl()] == 1, "GemJoin8/implementation-
           ↪ invalid");
        vat.slip(ilk, usr, int256(wad));
        require(gem.transferFrom(msg.sender, address(this), amt), "GemJoin8/
           ↪ failed-transfer");
        emit Join(usr, amt);
    }
```

9.10.9 exit(usr, amt) X

```
    function exit(address usr, uint256 amt) external {
        uint256 wad = mul(amt, 10 ** (18 - dec));
        require(int256(wad) >= 0, "GemJoin8/overflow");
        require(implementations[gem.erc20Impl()] == 1, "GemJoin8/implementation-
           ↪ invalid");
        vat.slip(ilk, msg.sender, -int256(wad));
        require(gem.transfer(usr, amt), "GemJoin8/failed-transfer");
        emit Exit(usr, amt);
    }
```

9.11 contract GemJoin9

```
// For a token that has a fee (PAXG)

contract GemJoin9 {
    // --- Data ---
    mapping (address => uint256) public wards; // Auth

    uint256 public live;                        // Active Flag
    uint256 public total;                       // Internal balance tracking

    VatLike public immutable vat;               // CDP Engine
    bytes32 public immutable ilk;               // Collateral Type
    GemLike public immutable gem;
    uint256 public immutable dec;

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event Join(address indexed usr, uint256 wad);
    event Exit(address indexed usr, uint256 wad);
    event Cage();
}
```

9.11.1 modifier auth()

```
    // --- Auth ---
    modifier auth {
        require(wards[msg.sender] == 1, "GemJoin9/not-authorized");
        _;
    }
```

9.11.2 constructor(vat_, ilk_, gem_) X

```
    constructor(address vat_, bytes32 ilk_, address gem_) public {
        vat = VatLike(vat_);
        ilk = ilk_;
        gem = GemLike(gem_);

        uint256 dec_ = GemLike(gem_).decimals();
        require(dec_ == 18, "GemJoin9/invalid-decimals");
        dec = dec_;

        live = 1;
        wards[msg.sender] = 1;
        emit Rely(msg.sender);
    }
```

9.11.3 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

9.11.4 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

9.11.5 cage() X a

```
    function cage() external auth {
        live = 0;
        emit Cage();
    }
```

9.11.6 _add(x, y)

```
    // --- Math ---
    function _add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "GemJoin9/overflow");
    }
```

9.11.7 _sub(x, y)

```
    function _sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "GemJoin9/underflow");
    }
```

9.11.8 join(usr) X

```
    // Allow dss-proxy-actions to send the gems with only 1 transfer
    // This should be called via token.transfer() followed by gemJoin.join()
       ↪ atomically or
    // someone else can steal your tokens
    function join(address usr) external returns (uint256 wad) {
        wad = _join(usr);

        emit Join(usr, wad);
    }
```

9.11.9 join(usr, wad) X

```
    function join(address usr, uint256 wad) external {
        require(gem.transferFrom(msg.sender, address(this), wad), "GemJoin9/
           ↪ failed-transfer");

        _join(usr);

        emit Join(usr, wad);
    }
```

9.11.10 _join(usr)

```
    function _join(address usr) internal returns (uint256 wad) {
        require(live == 1, "GemJoin9/not-live");

        uint256 _total = total;     // Cache to save an SLOAD
        wad = _sub(gem.balanceOf(address(this)), _total);
        require(int256(wad) >= 0, "GemJoin9/int256-overflow");

        total = _add(_total, wad);
        vat.slip(ilk, usr, int256(wad));
    }
```

9.11.11 exit(usr, wad) X

```
    function exit(address usr, uint256 wad) external {
        require(wad <= 2 ** 255, "GemJoin9/int256-overflow");

        total = _sub(total, wad);
        vat.slip(ilk, msg.sender, -int256(wad));

        require(gem.transfer(usr, wad), "GemJoin9/failed-transfer");
        emit Exit(usr, wad);
    }
```

9.12 contract AuthGemJoin

```
// For a token that needs restriction on the sources which are able to execute
    ↪ the join function (like SAI through Migration contract)

contract AuthGemJoin {
    VatLike public vat;
    bytes32 public ilk;
    GemLike public gem;
    uint256 public dec;
    uint256 public live;  // Access Flag

    // --- Auth ---
    mapping (address => uint256) public wards;

    // Events
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event Join(address indexed usr, uint256 wad);
    event Exit(address indexed usr, uint256 wad);
    event Cage();
}
```

9.12.1 modifier auth()

```
    modifier auth { require(wards[msg.sender] == 1, "AuthGemJoin/non-authed"); _
        ↪ ; }
```

9.12.2 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

9.12.3 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

9.12.4 constructor(vat_, ilk_, gem_) X

```
    constructor(address vat_, bytes32 ilk_, address gem_) public {
        wards[msg.sender] = 1;
        live = 1;
        vat = VatLike(vat_);
        ilk = ilk_;
        gem = GemLike(gem_);
        dec = gem.decimals();
        emit Rely(msg.sender);
    }
```

9.12.5 cage() X a

```
    function cage() external auth {
        live = 0;
        emit Cage();
    }
```

9.12.6 join(usr, wad) X a

```
    function join(address usr, uint256 wad) external auth {
        require(live == 1, "AuthGemJoin/not-live");
        require(int256(wad) >= 0, "AuthGemJoin/overflow");
        vat.slip(ilk, usr, int256(wad));
        require(gem.transferFrom(msg.sender, address(this), wad), "AuthGemJoin/
            ↪ failed-transfer");
        emit Join(usr, wad);
    }
```

9.12.7 exit(usr, wad) X

```
    function exit(address usr, uint256 wad) external {
        require(wad <= 2 ** 255, "AuthGemJoin/overflow");
        vat.slip(ilk, msg.sender, -int256(wad));
        require(gem.transfer(usr, wad), "AuthGemJoin/failed-transfer");
        emit Exit(usr, wad);
    }
```

9.13 contract ManagedGemJoin

```
// For a token that needs join/exit to be managed (like in permissioned vaults)

contract ManagedGemJoin {
    VatLike public immutable vat;
    bytes32 public immutable ilk;
    GemLike public immutable gem;
    uint256 public immutable dec;
    uint256 public live;  // Access Flag

    // --- Auth ---
    mapping (address => uint256) public wards;

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event Join(address indexed usr, uint256 amt);
    event Exit(address indexed urn, address indexed usr, uint256 amt);
    event Cage();
}
```

9.13.1 modifier auth()

```
    modifier auth { require(wards[msg.sender] == 1, "ManagedGemJoin/non-authed")
        ↪ ; _; }
```

9.13.2 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

9.13.3 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

9.13.4 constructor(vat_, ilk_, gem_) X

```
    constructor(address vat_, bytes32 ilk_, address gem_) public {
        live = 1;
        vat = VatLike(vat_);
        ilk = ilk_;
        gem = GemLike(gem_);

        uint256 dec_ = GemLike(gem_).decimals();
        require(dec_ <= 18, "ManagedGemJoin/decimals-19-or-higher");
        dec = dec_;

        wards[msg.sender] = 1;
        emit Rely(msg.sender);
    }
```

9.13.5 cage() X a

```
    function cage() external auth {
        live = 0;
        emit Cage();
    }
```

9.13.6 _mul(x, y)

```
    function _mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require(y == 0 || (z = x * y) / y == x, "ManagedGemJoin/overflow");
    }
```

9.13.7 join(usr, amt) X a

```
    function join(address usr, uint256 amt) external auth {
        require(live == 1, "ManagedGemJoin/not-live");
        uint256 wad = _mul(amt, 10 ** (18 - dec));
        require(wad <= (2 ** 255 - 1), "ManagedGemJoin/overflow");
        vat.slip(ilk, usr, int256(wad));
        require(gem.transferFrom(msg.sender, address(this), amt), "
            ↪ ManagedGemJoin/failed-transfer");
        emit Join(usr, amt);
    }
```

9.13.8 exit(urn, usr, amt) X a

```
    function exit(address urn, address usr, uint256 amt) external auth {
        uint256 wad = _mul(amt, 10 ** (18 - dec));
        require(wad <= 2 ** 255, "ManagedGemJoin/overflow");
        vat.slip(ilk, urn, -int256(wad));
        require(gem.transfer(usr, amt), "ManagedGemJoin/failed-transfer");
        emit Exit(urn, usr, amt);
    }
```

9.14 contract AAVE

```
contract AAVE {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "Aave Token";
    string  public  symbol = "AAVE";
    uint8   public  decimals = 18;
    uint256                                                 _supply;
    mapping (address => uint256)                            _balances;
    mapping (address => mapping (address => uint256))   _approvals;
}
```

9.14.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.14.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.14.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.14.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.14.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.14.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.14.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.14.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.14.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.15 contract BAL

```
contract BAL {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "Balancer";
    string  public  symbol = "BAL";
    uint8   public  decimals = 18;
    uint256                                                 _supply;
    mapping (address => uint256)                            _balances;
    mapping (address => mapping (address => uint256))   _approvals;
}
```

9.15.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.15.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.15.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.15.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.15.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.15.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.15.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.15.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.15.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.16 contract BAT

```
contract BAT {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "BAT";
    string  public  symbol = "BAT";
    uint256 public  decimals = 18;
    uint256                                             _supply;
    mapping (address => uint256)                        _balances;
    mapping (address => mapping (address => uint256))   _approvals;
}
```

9.16.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.16.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.16.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.16.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.16.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.16.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.16.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.16.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.16.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.17 contract COMP

```
contract COMP {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "COMP";
    string  public  symbol = "COMP";
    uint8   public  decimals = 18;
    uint256                                             _supply;
    mapping (address => uint256)                        _balances;
    mapping (address => mapping (address => uint256))   _approvals;
}
```

9.17.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.17.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.17.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.17.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.17.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.17.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.17.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.17.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.17.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.18 contract DGD

```
contract DGD {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "DGD";
    string  public  symbol = "DGD";
    uint256                                              _supply;
    mapping (address => uint256)                         _balances;
    mapping (address => mapping (address => uint256))   _approvals;
}
```

9.18.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.18.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.18.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.18.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.18.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.18.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.18.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.18.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.18.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.19 contract GNT

```
contract GNT {
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "GNT";
    string  public  symbol = "GNT";
    uint256 public  decimals = 18;
    uint256                        _supply;
    mapping (address => uint256)   _balances;
}
```

9.19.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.19.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.19.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.19.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.19.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.19.6 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        require(_balances[msg.sender] >= wad, "insufficient-balance");
        _balances[msg.sender] = sub(_balances[msg.sender], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(msg.sender, dst, wad);

        return true;
    }
```

9.20 contract GUSD

```
contract GUSD {
    event Approval(address indexed src, address indexed guy, uint wad);
    event Transfer(address indexed src, address indexed dst, uint wad);

    string  public  name = "GUSD";
    string  public  symbol = "GUSD";
    uint8   public  decimals = 2;
    address public  erc20Impl;
    uint256                                              _supply;
    mapping (address => uint256)                         _balances;
    mapping (address => mapping (address => uint256))   _approvals;
}
```

9.20.1 add(x, y)

```
    function add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.20.2 sub(x, y)

```
    function sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.20.3 constructor(supply) X

```
    constructor(uint supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
        setImplementation(address(this));
    }
```

9.20.4 setImplementation(newImplementation) X

```
    function setImplementation(address newImplementation) public {
        erc20Impl = newImplementation;
    }
```

9.20.5 totalSupply()

```
    function totalSupply() public view returns (uint) {
        return _supply;
    }
```

9.20.6 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint) {
        return _balances[src];
    }
```

9.20.7 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint) {
        return _approvals[src][guy];
    }
```

9.20.8 transfer(dst, wad) X

```
    function transfer(address dst, uint wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.20.9 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.20.10 approve(guy, wad) X

```
    function approve(address guy, uint wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.21 contract KNC

```
contract KNC {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "KNC";
    string  public  symbol = "KNC";
    uint256 public  decimals = 18;
    uint256                                              _supply;
    mapping (address => uint256)                         _balances;
    mapping (address => mapping (address => uint256))   _approvals;
}
```

9.21.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.21.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.21.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.21.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.21.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.21.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.21.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.21.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.21.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.22 contract LINK

```
contract LINK {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "LINK";
    string  public  symbol = "LINK";
    uint8   public  decimals = 18;
    uint256                                              _supply;
    mapping (address => uint256)                         _balances;
    mapping (address => mapping (address => uint256))   _approvals;
}
```

9.22.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.22.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.22.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.22.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.22.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.22.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.22.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.22.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.22.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.23 contract LRC

```
contract LRC {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "LRC";
    string  public  symbol = "LRC";
    uint8   public  decimals = 18;
    uint256                                                 _supply;
    mapping (address => uint256)                            _balances;
    mapping (address => mapping (address => uint256))       _approvals;
}
```

9.23.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.23.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.23.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.23.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.23.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.23.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.23.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.23.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.23.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.24 contract MANA

```
contract MANA {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "MANA";
    string  public  symbol = "MANA";
    uint8   public  decimals = 18;
    uint256                                                  _supply;
    mapping (address => uint256)                             _balances;
    mapping (address => mapping (address => uint256))        _approvals;
}
```

9.24.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.24.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.24.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.24.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.24.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.24.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.24.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.24.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.24.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.25 contract MATIC

```
contract MATIC {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "MATIC";
    string  public  symbol = "MATIC";
    uint256 public  decimals = 18;
    uint256                                              _supply;
    mapping (address => uint256)                         _balances;
    mapping (address => mapping (address => uint256))   _approvals;
}
```

9.25.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.25.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.25.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.25.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.25.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.25.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.25.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.25.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.25.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.26 contract OMG

```
contract OMG {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "OMG";
    string  public  symbol = "OMG";
    uint256 public  decimals = 18;
    uint256                                                  _supply;
    mapping (address => uint256)                             _balances;
    mapping (address => mapping (address => uint256))   _approvals;
}
```

9.26.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.26.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.26.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.26.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.26.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.26.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.26.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public {
        transferFrom(msg.sender, dst, wad);
    }
```

9.26.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad) public
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);
    }
```

9.26.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);
    }
```

9.27 contract PAXG

```
contract PAXG {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    event FeeCollected(address indexed from, address indexed to, uint256 value);
    event FeeRateSet(
        uint256 indexed oldFeeRate,
        uint256 indexed newFeeRate
    );

    string  public  name = "Paxos Gold";
    string  public  symbol = "PAXG";
    uint256 public  decimals = 18;
    uint256                                           _supply;
    mapping (address => uint256)                      _balances;
    mapping (address => mapping (address => uint256))  _approvals;

    uint256 public constant feeParts = 1000000;
    uint256 public feeRate;
    address public feeRecipient = 0x57aAeAE905376a4B1899bA81364b4cE2519CBfB3;
       ↪       // Doesn't really matter where the fees go (send to faucet)

}
```

9.27.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.27.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.27.3 mul(a, b)

```
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b);

        return c;
    }
```

9.27.4 div(a, b)

```
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b > 0);
        uint256 c = a / b;

        return c;
    }
```

9.27.5 constructor(supply, fee) X

```
    constructor(uint256 supply, uint256 fee) public {
        _balances[msg.sender] = supply;
        _supply = supply;

        setFeeRate(fee);
    }
```

9.27.6 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.27.7 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.27.8 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.27.9 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.27.10 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        uint256 _fee = getFeeFor(wad);
        uint256 _principal = sub(wad, _fee);
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], _principal);

        emit Transfer(src, dst, _principal);
        emit Transfer(src, feeRecipient, _fee);

        if (_fee > 0) {
            _balances[feeRecipient] = add(_balances[feeRecipient], _fee);
            emit FeeCollected(src, feeRecipient, _fee);
        }

        return true;
    }
```

9.27.11 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.27.12 setFeeRate(_newFeeRate)

```
    function setFeeRate(uint256 _newFeeRate) internal {
        require(_newFeeRate <= feeParts, "cannot set fee rate above 100%");
        uint256 _oldFeeRate = feeRate;
        feeRate = _newFeeRate;
        emit FeeRateSet(_oldFeeRate, feeRate);
    }
```

9.27.13 getFeeFor(_value)

```
    function getFeeFor(uint256 _value) public view returns (uint256) {
        if (feeRate == 0) {
            return 0;
        }

        return div(mul(_value, feeRate), feeParts);
    }
```

9.28 contract PAXUSD

```
contract PAXUSD {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "PAXUSD";
    string  public  symbol = "PAXUSD";
    uint256 public  decimals = 18;
    uint256                                              _supply;
    mapping (address => uint256)                         _balances;
    mapping (address => mapping (address => uint256))   _approvals;
}
```

9.28.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.28.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.28.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.28.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.28.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.28.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.28.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.28.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.28.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.29 contract RENBTC

```
contract RENBTC {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "RENBTC";
    string  public  symbol = "RENBTC";
    uint256 public  decimals = 8;
    uint256                                                 _supply;
    mapping (address => uint256)                            _balances;
    mapping (address => mapping (address => uint256))       _approvals;
}
```

9.29.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.29.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.29.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.29.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.29.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.29.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.29.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.29.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.29.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.30 contract REP

```
contract REP {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "REP";
    string  public  symbol = "REP";
    uint256 public  decimals = 18;
    uint256                                                _supply;
    mapping (address => uint256)                           _balances;
    mapping (address => mapping (address => uint256))      _approvals;
}
```

9.30.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.30.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.30.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.30.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.30.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.30.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.30.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.30.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.30.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.31 contract TUSD

```
contract TUSD {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "TrueUSD";
    string  public  symbol = "TUSD";
    uint8   public  decimals = 18;
    address public  implementation;
    uint256                                             _supply;
    mapping (address => uint256)                        _balances;
    mapping (address => mapping (address => uint256))   _approvals;
}
```

9.31.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.31.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.31.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
        setImplementation(address(this));
    }
```

9.31.4 setImplementation(newImplementation) X

```
    function setImplementation(address newImplementation) public {
        implementation = newImplementation;
    }
```

9.31.5 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.31.6 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.31.7 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.31.8 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.31.9 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
               ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.31.10 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.32 contract UNI

```
contract UNI {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "Uniswap";
    string  public  symbol = "UNI";
    uint8   public  decimals = 18;
    uint256                                                 _supply;
    mapping (address => uint256)                            _balances;
    mapping (address => mapping (address => uint256))       _approvals;
}
```

9.32.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.32.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.32.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.32.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.32.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.32.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.32.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.32.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
               ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.32.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.33 contract USDC

```
contract USDC {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "USDC";
    string  public  symbol = "USDC";
    uint256 public  decimals = 6;
    uint256                                              _supply;
    mapping (address => uint256)                         _balances;
    mapping (address => mapping (address => uint256))   _approvals;
}
```

9.33.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.33.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.33.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.33.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.33.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.33.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.33.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.33.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.33.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.34 contract USDT

```
contract USDT {
    using SafeMath for uint256;

    string  public name = "Tether";
    string  public symbol = "USDT";
    uint256 public decimals = 6;
    address public upgradedAddress;
    bool    public deprecated;

    mapping (address => mapping (address => uint256)) public allowed;
    mapping (address => uint256) public balances;

    uint256 public constant MAX_UINT = 2**256 - 1;

    address public owner;
    uint256 public basisPointsRate;
    uint256 public maximumFee;

    event Approval(address indexed owner, address indexed spender, uint256 value
       ↪ );
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Deprecate(address newAddress);
}
```

9.34.1 modifier onlyPayloadSize(size)

```
    modifier onlyPayloadSize(uint256 size) {
        require(!(msg.data.length < size + 4));
        _;
    }
```

9.34.2 constructor(_initialSupply) X

```
    constructor(uint256 _initialSupply) public {
        balances[msg.sender] = _initialSupply;
    }
```

9.34.3 changeFees(_basisPointsRate, _maximumFee) X

```
    function changeFees(uint256 _basisPointsRate, uint256 _maximumFee) public {
        basisPointsRate = _basisPointsRate;
        maximumFee = _maximumFee;
    }
```

9.34.4 balanceOf(_owner)

```
    function balanceOf(address _owner) public view returns (uint256 balance) {
        return balances[_owner];
    }
```

9.34.5 transfer(_to, _value) X

```
    function transfer(address _to, uint256 _value) public onlyPayloadSize(2 *
       ↪ 32) {
        uint256 fee = (_value.mul(basisPointsRate)).div(10000);
        if (fee > maximumFee) {
            fee = maximumFee;
        }
        uint256 sendAmount = _value.sub(fee);
        balances[msg.sender] = balances[msg.sender].sub(_value);
        balances[_to] = balances[_to].add(sendAmount);
        if (fee > 0) {
            balances[owner] = balances[owner].add(fee);
            emit Transfer(msg.sender, owner, fee);
        }
        emit Transfer(msg.sender, _to, sendAmount);
    }
```

9.34.6 transferFrom(_from, _to, _value) X

```
    function transferFrom(address _from, address _to, uint256 _value) public
        ↪ onlyPayloadSize(3 * 32) {
        uint256 _allowance = allowed[_from][msg.sender];
        // Check is not needed because sub(_allowance, _value) will already
            ↪ throw if this condition is not met
        // if (_value > _allowance) throw;
        uint256 fee = (_value.mul(basisPointsRate)).div(10000);
        if (fee > maximumFee) {
            fee = maximumFee;
        }
        if (_allowance < MAX_UINT) {
            allowed[_from][msg.sender] = _allowance.sub(_value);
        }
        uint256 sendAmount = _value.sub(fee);
        balances[_from] = balances[_from].sub(_value);
        balances[_to] = balances[_to].add(sendAmount);
        if (fee > 0) {
            balances[owner] = balances[owner].add(fee);
            emit Transfer(_from, owner, fee);
        }
        emit Transfer(_from, _to, sendAmount);
    }
```

9.34.7 approve(_spender, _value) X

```
    function approve(address _spender, uint256 _value) public onlyPayloadSize(2
        ↪ * 32) {
        // To change the approve amount you first have to reduce the addresses‘
        // allowance to zero by calling ‘approve(_spender, 0)‘ if it is not
        // already 0 to mitigate the race condition described here:
        // https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
        require(!((_value != 0) && (allowed[msg.sender][_spender] != 0)));
        allowed[msg.sender][_spender] = _value;
        emit Approval(msg.sender, _spender, _value);
    }
```

9.34.8 allowance(_owner, _spender)

```
    function allowance(address _owner, address _spender) public view returns (
        ↪ uint256 remaining) {
        return allowed[_owner][_spender];
    }
```

9.34.9 deprecate(_upgradedAddress) X

```
    function deprecate(address _upgradedAddress) public {
        deprecated = true;
        upgradedAddress = _upgradedAddress;
        emit Deprecate(_upgradedAddress);
    }
```

9.35 contract WBTC

```
contract WBTC {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "WBTC";
    string  public  symbol = "WBTC";
    uint256 public  decimals = 8;
    uint256                                              _supply;
    mapping (address => uint256)                         _balances;
    mapping (address => mapping (address => uint256))   _approvals;
}
```

9.35.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.35.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.35.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.35.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.35.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.35.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.35.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.35.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.35.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.36 contract WSTETH

```
contract WSTETH {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "Wrapped liquid staked Ether 2.0";
    string  public  symbol = "WSTETH";
    uint8   public  decimals = 18;
    uint256                                             _supply;
    mapping (address => uint256)                        _balances;
    mapping (address => mapping (address => uint256))  _approvals;
}
```

9.36.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.36.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.36.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.36.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.36.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.36.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.36.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.36.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.36.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.37 contract YFI

```
contract YFI {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "yearn.finance";
    string  public  symbol = "YFI";
    uint8   public  decimals = 18;
    uint256                                              _supply;
    mapping (address => uint256)                         _balances;
    mapping (address => mapping (address => uint256))   _approvals;
}
```

9.37.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.37.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.37.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.37.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.37.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.37.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.37.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.37.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
               ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.37.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

9.38 contract ZRX

```
contract ZRX {
    event Approval(address indexed src, address indexed guy, uint256 wad);
    event Transfer(address indexed src, address indexed dst, uint256 wad);

    string  public  name = "ZRX";
    string  public  symbol = "ZRX";
    uint256 public  decimals = 18;
    uint256                                              _supply;
    mapping (address => uint256)                         _balances;
    mapping (address => mapping (address => uint256))   _approvals;
}
```

9.38.1 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

9.38.2 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

9.38.3 constructor(supply) X

```
    constructor(uint256 supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }
```

9.38.4 totalSupply()

```
    function totalSupply() public view returns (uint256) {
        return _supply;
    }
```

9.38.5 balanceOf(src)

```
    function balanceOf(address src) public view returns (uint256) {
        return _balances[src];
    }
```

9.38.6 allowance(src, guy)

```
    function allowance(address src, address guy) public view returns (uint256) {
        return _approvals[src][guy];
    }
```

9.38.7 transfer(dst, wad) X

```
    function transfer(address dst, uint256 wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

9.38.8 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint256 wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "insufficient-approval")
                ↪ ;
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
```

9.38.9 approve(guy, wad) X

```
    function approve(address guy, uint256 wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true;
    }
```

Chapter 10

Peg Stability

10.1 contract AuthGemJoin5

```
// Authed GemJoin for a token that has a lower precision than 18 and it has
    ↪ decimals (like USDC)

contract AuthGemJoin5 {
    // --- Auth ---
    mapping (address => uint256) public wards;

    VatLike public immutable vat;
    bytes32 public immutable ilk;
    GemLike public immutable gem;
    uint256 public immutable dec;
    uint256 public live;  // Access Flag

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event Cage();
    event Join(address indexed urn, uint256 amt, address indexed msgSender);
    event Exit(address indexed usr, uint256 amt);
}
```

10.1.1 modifier auth()

```
    modifier auth { require(wards[msg.sender] == 1); _; }
```

10.1.2 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

10.1.3 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

10.1.4 constructor(vat_, ilk_, gem_) X

```
    constructor(address vat_, bytes32 ilk_, address gem_) public {
        gem = GemLike(gem_);
        uint256 dec_ = dec = GemLike(gem_).decimals();
        require(dec_ < 18, "AuthGemJoin5/decimals-18-or-higher");
        wards[msg.sender] = 1;
        emit Rely(msg.sender);
        live = 1;
        vat = VatLike(vat_);
        ilk = ilk_;
    }
```

10.1.5 cage() X a

```
    function cage() external auth {
        live = 0;
        emit Cage();
    }
```

10.1.6 mul(x, y)

```
    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require(y == 0 || (z = x * y) / y == x, "AuthGemJoin5/overflow");
    }
```

10.1.7 join(urn, amt, msgSender) X a

```
    function join(address urn, uint256 amt, address msgSender) external auth {
        require(live == 1, "AuthGemJoin5/not-live");
        uint256 wad = mul(amt, 10 ** (18 - dec));
        require(int256(wad) >= 0, "AuthGemJoin5/overflow");
        vat.slip(ilk, urn, int256(wad));
        require(gem.transferFrom(msgSender, address(this), amt), "AuthGemJoin5/
           ↪ failed-transfer");
        emit Join(urn, amt, msgSender);
    }
```

10.1.8 exit(usr, amt) X

```
    function exit(address usr, uint256 amt) external {
        uint256 wad = mul(amt, 10 ** (18 - dec));
        require(int256(wad) >= 0, "AuthGemJoin5/overflow");
        vat.slip(ilk, msg.sender, -int256(wad));
        require(gem.transfer(usr, amt), "AuthGemJoin5/failed-transfer");
        emit Exit(usr, amt);
    }
```

10.2 contract AuthGemJoin8

```
// AuthGemJoin8
// For a token that has a lower precision than 18, has decimals and it is
   ↪ upgradable (like GUSD)

contract AuthGemJoin8 {
    // --- Auth ---
    mapping (address => uint256) public wards;

    VatLike public immutable vat;
    bytes32 public immutable ilk;
    GemLike public immutable gem;
    uint256 public immutable dec;
    uint256 public live;  // Access Flag

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event Cage();
    event Join(address indexed urn, uint256 amt, address indexed msgSender);
    event Exit(address indexed usr, uint256 amt);
    event SetImplementation(address indexed implementation, uint256 permitted);

    mapping (address => uint256) public implementations;
}
```

10.2.1 modifier auth()

```
    modifier auth { require(wards[msg.sender] == 1); _; }
```

10.2.2 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

10.2.3 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

10.2.4 constructor(vat_, ilk_, gem_) X

```
    constructor(address vat_, bytes32 ilk_, address gem_) public {
        gem = GemLike(gem_);
        uint256 dec_ = dec = GemLike(gem_).decimals();
        require(dec_ < 18, "AuthGemJoin8/decimals-18-or-higher");
        wards[msg.sender] = 1;
        emit Rely(msg.sender);
        live = 1;
        setImplementation(GemLike(gem_).erc20Impl(), 1);
        vat = VatLike(vat_);
        ilk = ilk_;
    }
```

10.2.5 cage() X a

```
    function cage() external auth {
        live = 0;
        emit Cage();
    }
```

10.2.6 setImplementation(implementation, permitted) X a

```
    function setImplementation(address implementation, uint256 permitted) public
       ↪  auth {
        implementations[implementation] = permitted;  // 1 live, 0 disable
        emit SetImplementation(implementation, permitted);
    }
```

10.2.7 mul(x, y)

```
    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require(y == 0 || (z = x * y) / y == x, "AuthGemJoin8/overflow");
    }
```

10.2.8 join(urn, amt, msgSender) X a

```
    function join(address urn, uint256 amt, address msgSender) external auth {
        require(live == 1, "AuthGemJoin8/not-live");
        uint256 wad = mul(amt, 10 ** (18 - dec));
        require(int256(wad) >= 0, "AuthGemJoin8/overflow");
        require(implementations[gem.erc20Impl()] == 1, "AuthGemJoin8/
           ↪ implementation-invalid");
        vat.slip(ilk, urn, int256(wad));
        require(gem.transferFrom(msgSender, address(this), amt), "AuthGemJoin8/
           ↪ failed-transfer");
        emit Join(urn, amt, msgSender);
    }
```

10.2.9 exit(usr, amt) X

```
    function exit(address usr, uint256 amt) external {
        uint256 wad = mul(amt, 10 ** (18 - dec));
        require(int256(wad) >= 0, "AuthGemJoin8/overflow");
        require(implementations[gem.erc20Impl()] == 1, "AuthGemJoin8/
           ↪ implementation-invalid");
        vat.slip(ilk, msg.sender, -int256(wad));
        require(gem.transfer(usr, amt), "AuthGemJoin8/failed-transfer");
        emit Exit(usr, amt);
    }
```

10.3 contract AuthGemJoin(2)

```
contract AuthGemJoin {
    // --- Auth ---
    mapping (address => uint) public wards;

    VatLike public immutable vat;   // CDP Engine
    bytes32 public immutable ilk;   // Collateral Type
    GemLike public immutable gem;
    uint    public immutable dec;
    uint    public live;  // Active Flag

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event Cage();
    event Join(address indexed urn, uint256 wad, address indexed msgSender);
    event Exit(address indexed guy, uint256 wad);
}
```

10.3.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "GemJoin/not-authorized");
        _;
    }
```

10.3.2 rely(usr) X a

```
    function rely(address usr) external auth { wards[usr] = 1; emit Rely(usr); }
```

10.3.3 deny(usr) X a

```
    function deny(address usr) external auth { wards[usr] = 0; emit Deny(usr); }
```

10.3.4 constructor(vat_, ilk_, gem_) X

```
    constructor(address vat_, bytes32 ilk_, address gem_) public {
        wards[msg.sender] = 1;
        emit Rely(msg.sender);
        live = 1;
        vat = VatLike(vat_);
        ilk = ilk_;
        gem = GemLike(gem_);
        dec = GemLike(gem_).decimals();
    }
```

10.3.5 cage() X a

```
    function cage() external auth {
        live = 0;
        emit Cage();
    }
```

10.3.6 join(urn, wad, msgSender) X a

```
    function join(address urn, uint wad, address msgSender) external auth {
        require(live == 1, "GemJoin/not-live");
        require(int(wad) >= 0, "GemJoin/overflow");
        vat.slip(ilk, urn, int(wad));
        require(gem.transferFrom(msgSender, address(this), wad), "GemJoin/failed
            ↪ -transfer");
        emit Join(urn, wad, msgSender);
    }
```

10.3.7 exit(guy, wad) X

```
    function exit(address guy, uint wad) external {
        require(wad <= 2 ** 255, "GemJoin/overflow");
        vat.slip(ilk, msg.sender, -int(wad));
        require(gem.transfer(guy, wad), "GemJoin/failed-transfer");
        emit Exit(guy, wad);
    }
```

10.4 contract DssPsm

```
// Peg Stability Module
// Allows anyone to go between Dai and the Gem by pooling the liquidity
// An optional fee is charged for incoming and outgoing transfers

contract DssPsm {

    // --- Auth ---
    mapping (address => uint256) public wards;

    VatAbstract immutable public vat;
    AuthGemJoinAbstract immutable public gemJoin;
    DaiAbstract immutable public dai;
    DaiJoinAbstract immutable public daiJoin;
    bytes32 immutable public ilk;
    address immutable public vow;

    uint256 immutable internal to18ConversionFactor;

    uint256 public tin;         // toll in [wad]
    uint256 public tout;        // toll out [wad]

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event File(bytes32 indexed what, uint256 data);
    event SellGem(address indexed owner, uint256 value, uint256 fee);
    event BuyGem(address indexed owner, uint256 value, uint256 fee);

    // --- Math ---
    uint256 constant WAD = 10 ** 18;
    uint256 constant RAY = 10 ** 27;

}
```

10.4.1 modifier auth()

```
    modifier auth { require(wards[msg.sender] == 1); _; }
```

10.4.2 rely(usr) X a

```
    function rely(address usr) external auth { wards[usr] = 1; emit Rely(usr); }
```

10.4.3 deny(usr) X a

```
    function deny(address usr) external auth { wards[usr] = 0; emit Deny(usr); }
```

10.4.4 constructor(gemJoin_, daiJoin_, vow_) X

```
    // --- Init ---
    constructor(address gemJoin_, address daiJoin_, address vow_) public {
        wards[msg.sender] = 1;
        emit Rely(msg.sender);
        AuthGemJoinAbstract gemJoin__ = gemJoin = AuthGemJoinAbstract(gemJoin_);
        DaiJoinAbstract daiJoin__ = daiJoin = DaiJoinAbstract(daiJoin_);
        VatAbstract vat__ = vat = VatAbstract(address(gemJoin__.vat()));
        DaiAbstract dai__ = dai = DaiAbstract(address(daiJoin__.dai()));
        ilk = gemJoin__.ilk();
        vow = vow_;
        to18ConversionFactor = 10 ** (18 - gemJoin__.dec());
        dai__.approve(daiJoin_, uint256(-1));
        vat__.hope(daiJoin_);
    }
```

10.4.5 add(x, y)

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x);
    }
```

10.4.6 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x);
    }
```

10.4.7 mul(x, y)

```
    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require(y == 0 || (z = x * y) / y == x);
    }
```

10.4.8 file(what, data) X a

```
    // --- Administration ---
    function file(bytes32 what, uint256 data) external auth {
        if (what == "tin") tin = data;
        else if (what == "tout") tout = data;
        else revert("DssPsm/file-unrecognized-param");

        emit File(what, data);
    }
```

10.4.9 hope(usr) X a

```
    // hope can be used to transfer control of the PSM vault to another contract
    // This can be used to upgrade the contract
    function hope(address usr) external auth {
        vat.hope(usr);
    }
```

10.4.10 nope(usr) X a

```
    function nope(address usr) external auth {
        vat.nope(usr);
    }
```

10.4.11 sellGem(usr, gemAmt) X

```
    // --- Primary Functions ---
    function sellGem(address usr, uint256 gemAmt) external {
        uint256 gemAmt18 = mul(gemAmt, to18ConversionFactor);
        uint256 fee = mul(gemAmt18, tin) / WAD;
        uint256 daiAmt = sub(gemAmt18, fee);
        gemJoin.join(address(this), gemAmt, msg.sender);
        vat.frob(ilk, address(this), address(this), address(this), int256(
            ↪ gemAmt18), int256(gemAmt18));
        vat.move(address(this), vow, mul(fee, RAY));
        daiJoin.exit(usr, daiAmt);
        emit SellGem(usr, gemAmt, fee);
    }
```

10.4.12 buyGem(usr, gemAmt) X

```
    function buyGem(address usr, uint256 gemAmt) external {
        uint256 gemAmt18 = mul(gemAmt, to18ConversionFactor);
        uint256 fee = mul(gemAmt18, tout) / WAD;
        uint256 daiAmt = add(gemAmt18, fee);
        require(dai.transferFrom(msg.sender, address(this), daiAmt), "DssPsm/
            ↪ failed-transfer");
        daiJoin.join(address(this), daiAmt);
        vat.frob(ilk, address(this), address(this), address(this), -int256(
            ↪ gemAmt18), -int256(gemAmt18));
        gemJoin.exit(usr, gemAmt);
        vat.move(address(this), vow, mul(fee, RAY));

        emit BuyGem(usr, gemAmt, fee);
    }
```

Chapter 11

WETH

11.1 contract DSWethFactory

```
contract DSWethFactory {
    event LogMake(address indexed creator, address token);
}
```

11.1.1 make() X

```
    function make() public returns (WETH9_ result) {
        result = new WETH9_();
        emit LogMake(msg.sender, address(result));
    }
```

11.2 contract WETH

```
contract WETH is ERC20, WETHEvents {
    function join() public payable;
    function exit(uint wad) public;
}
```

Inherited:

```
contract WETHEvents is ERC20Events {
    event Join(address indexed dst, uint wad);
    event Exit(address indexed src, uint wad);
}
```

```
contract ERC20 is ERC20Events {
    function totalSupply() public view returns (uint);
    function balanceOf(address guy) public view returns (uint);
    function allowance(address src, address guy) public view returns (uint);

    function approve(address guy, uint wad) public returns (bool);
    function transfer(address dst, uint wad) public returns (bool);
    function transferFrom(
        address src, address dst, uint wad
    ) public returns (bool);
}
```

```
contract ERC20Events {
    event Approval(address indexed src, address indexed guy, uint wad);
    event Transfer(address indexed src, address indexed dst, uint wad);
}
```

11.3 contract WETH9_

```
contract WETH9_ {
    string public name     = "Wrapped Ether";
    string public symbol   = "WETH";
    uint8  public decimals = 18;

    event  Approval(address indexed src, address indexed guy, uint wad);
    event  Transfer(address indexed src, address indexed dst, uint wad);
    event  Deposit(address indexed dst, uint wad);
    event  Withdrawal(address indexed src, uint wad);

    mapping (address => uint)                       public  balanceOf;
    mapping (address => mapping (address => uint))  public  allowance;
}
```

11.3.1 fallback() X

```
    function() external payable {
        deposit();
    }
```

11.3.2 deposit() X

```
    function deposit() public payable {
        balanceOf[msg.sender] += msg.value;
        emit Deposit(msg.sender, msg.value);
    }
```

11.3.3 withdraw(wad) X

```
    function withdraw(uint wad) public {
        require(balanceOf[msg.sender] >= wad);
        balanceOf[msg.sender] -= wad;
        msg.sender.transfer(wad);
        emit Withdrawal(msg.sender, wad);
    }
```

11.3.4 totalSupply()

```
    function totalSupply() public view returns (uint) {
        return address(this).balance;
    }
```

11.3.5 approve(guy, wad) X

```
    function approve(address guy, uint wad) public returns (bool) {
        allowance[msg.sender][guy] = wad;
        emit Approval(msg.sender, guy, wad);
        return true;
    }
```

11.3.6 transfer(dst, wad) X

```
    function transfer(address dst, uint wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }
```

11.3.7 transferFrom(src, dst, wad) X

```
    function transferFrom(address src, address dst, uint wad)
        public
        returns (bool)
    {
        require(balanceOf[src] >= wad);

        if (src != msg.sender && allowance[src][msg.sender] != uint(-1)) {
            require(allowance[src][msg.sender] >= wad);
            allowance[src][msg.sender] -= wad;
        }

        balanceOf[src] -= wad;
        balanceOf[dst] += wad;

        emit Transfer(src, dst, wad);

        return true;
    }
```

Chapter 12

Proxy

12.1 contract UrnHandler

```
contract UrnHandler {
}
```

12.1.1 constructor(vat) X

```
    constructor(address vat) public {
        VatLike(vat).hope(msg.sender);
    }
```

12.2 contract DssCdpManager

```
contract DssCdpManager {
    address                      public vat;
    uint                         public cdpi;       // Auto incremental
    mapping (uint => address)    public urns;       // CDPId => UrnHandler
    mapping (uint => List)       public list;       // CDPId => Prev & Next CDPIds (
        ↪ double linked list)
    mapping (uint => address)    public owns;       // CDPId => Owner
    mapping (uint => bytes32)    public ilks;       // CDPId => Ilk

    mapping (address => uint)    public first;      // Owner => First CDPId
    mapping (address => uint)    public last;       // Owner => Last CDPId
    mapping (address => uint)    public count;      // Owner => Amount of CDPs

    mapping (
        address => mapping (
            uint => mapping (
                address => uint
            )
        )
    ) public cdpCan;                                // Owner => CDPId => Allowed
        ↪ Addr => True/False

    mapping (
        address => mapping (
            address => uint
        )
    ) public urnCan;                                // Urn => Allowed Addr => True/
        ↪ False

    event NewCdp(address indexed usr, address indexed own, uint indexed cdp);
}
```

12.2.1 struct DssCdpManager.List

```
    struct List {
        uint prev;
        uint next;
    }
```

12.2.2 modifier cdpAllowed(cdp)

```
    modifier cdpAllowed(
        uint cdp
    ) {
        require(msg.sender == owns[cdp] || cdpCan[owns[cdp]][cdp][msg.sender] ==
            ↪  1, "cdp-not-allowed");
        _;
    }
```

12.2.3 modifier urnAllowed(urn)

```
    modifier urnAllowed(
        address urn
    ) {
        require(msg.sender == urn || urnCan[urn][msg.sender] == 1, "urn-not-
            ↪ allowed");
        _;
    }
```

12.2.4 constructor(vat_) X

```
    constructor(address vat_) public {
        vat = vat_;
    }
```

12.2.5 add(x, y)

```
    function add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x);
    }
```

12.2.6 sub(x, y)

```
    function sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x);
    }
```

12.2.7 toInt(x)

```
    function toInt(uint x) internal pure returns (int y) {
        y = int(x);
        require(y >= 0);
    }
```

12.2.8 cdpAllow(cdp, usr, ok) X

```
    // Allow/disallow a usr address to manage the cdp.
    function cdpAllow(
        uint cdp,
        address usr,
        uint ok
    ) public cdpAllowed(cdp) {
        cdpCan[owns[cdp]][cdp][usr] = ok;
    }
```

12.2.9 urnAllow(usr, ok) X

```
    // Allow/disallow a usr address to quit to the the sender urn.
    function urnAllow(
        address usr,
        uint ok
    ) public {
        urnCan[msg.sender][usr] = ok;
    }
```

12.2.10 open(ilk, usr) X

```
    // Open a new cdp for a given usr address.
    function open(
        bytes32 ilk,
        address usr
    ) public returns (uint) {
        require(usr != address(0), "usr-address-0");

        cdpi = add(cdpi, 1);
        urns[cdpi] = address(new UrnHandler(vat));
        owns[cdpi] = usr;
        ilks[cdpi] = ilk;
        // Add new CDP to double linked list and pointers
        if (first[usr] == 0) {
            first[usr] = cdpi;
        }
        if (last[usr] != 0) {
            list[cdpi].prev = last[usr];
            list[last[usr]].next = cdpi;
        }
        last[usr] = cdpi;
        count[usr] = add(count[usr], 1);

        emit NewCdp(msg.sender, usr, cdpi);
        return cdpi;
    }
```

12.2.11 give(cdp, dst) X

```
    // Give the cdp ownership to a dst address.
    function give(
        uint cdp,
        address dst
    ) public cdpAllowed(cdp) {
        require(dst != address(0), "dst-address-0");
        require(dst != owns[cdp], "dst-already-owner");

        // Remove transferred CDP from double linked list of origin user and
            ↪ pointers
        if (list[cdp].prev != 0) {
            list[list[cdp].prev].next = list[cdp].next;         // Set the next
                ↪ pointer of the prev cdp (if exists) to the next of the
                ↪ transferred one
        }
        if (list[cdp].next != 0) {                               // If wasn't the
            ↪  last one
            list[list[cdp].next].prev = list[cdp].prev;         // Set the prev
                ↪ pointer of the next cdp to the prev of the transferred one
        } else {                                                 // If was the
            ↪ last one
            last[owns[cdp]] = list[cdp].prev;                    // Update last
                ↪ pointer of the owner
        }
        if (first[owns[cdp]] == cdp) {                           // If was the
            ↪ first one
            first[owns[cdp]] = list[cdp].next;                   // Update first
                ↪ pointer of the owner
        }
        count[owns[cdp]] = sub(count[owns[cdp]], 1);

        // Transfer ownership
        owns[cdp] = dst;

        // Add transferred CDP to double linked list of destiny user and
            ↪ pointers
        list[cdp].prev = last[dst];
        list[cdp].next = 0;
        if (last[dst] != 0) {
            list[last[dst]].next = cdp;
        }
        if (first[dst] == 0) {
            first[dst] = cdp;
        }
        last[dst] = cdp;
        count[dst] = add(count[dst], 1);
    }
```

12.2.12 frob(cdp, dink, dart) X

```
    // Frob the cdp keeping the generated DAI or collateral freed in the cdp urn
       ↪  address.
    function frob(
        uint cdp,
        int dink,
        int dart
    ) public cdpAllowed(cdp) {
        address urn = urns[cdp];
        VatLike(vat).frob(
            ilks[cdp],
            urn,
            urn,
            urn,
            dink,
            dart
        );
    }
```

12.2.13 flux(cdp, dst, wad) X

```
    // Transfer wad amount of cdp collateral from the cdp address to a dst
       ↪ address.
    function flux(
        uint cdp,
        address dst,
        uint wad
    ) public cdpAllowed(cdp) {
        VatLike(vat).flux(ilks[cdp], urns[cdp], dst, wad);
    }
```

12.2.14 flux(ilk, cdp, dst, wad) X

```
    // Transfer wad amount of any type of collateral (ilk) from the cdp address
       ↪ to a dst address.
    // This function has the purpose to take away collateral from the system
       ↪ that doesn't correspond to the cdp but was sent there wrongly.
    function flux(
        bytes32 ilk,
        uint cdp,
        address dst,
        uint wad
    ) public cdpAllowed(cdp) {
        VatLike(vat).flux(ilk, urns[cdp], dst, wad);
    }
```

12.2.15 move(cdp, dst, rad) X

```
    // Transfer wad amount of DAI from the cdp address to a dst address.
    function move(
        uint cdp,
        address dst,
        uint rad
    ) public cdpAllowed(cdp) {
        VatLike(vat).move(urns[cdp], dst, rad);
    }
```

12.2.16 quit(cdp, dst) X

```
    // Quit the system, migrating the cdp (ink, art) to a different dst urn
    function quit(
        uint cdp,
        address dst
    ) public cdpAllowed(cdp) urnAllowed(dst) {
        (uint ink, uint art) = VatLike(vat).urns(ilks[cdp], urns[cdp]);
        VatLike(vat).fork(
            ilks[cdp],
            urns[cdp],
            dst,
            toInt(ink),
            toInt(art)
        );
    }
```

12.2.17 enter(src, cdp) X

```
    // Import a position from src urn to the urn owned by cdp
    function enter(
        address src,
        uint cdp
    ) public urnAllowed(src) cdpAllowed(cdp) {
        (uint ink, uint art) = VatLike(vat).urns(ilks[cdp], src);
        VatLike(vat).fork(
            ilks[cdp],
            src,
            urns[cdp],
            toInt(ink),
            toInt(art)
        );
    }
```

12.2.18 shift(cdpSrc, cdpDst) X

```
    // Move a position from cdpSrc urn to the cdpDst urn
    function shift(
        uint cdpSrc,
        uint cdpDst
    ) public cdpAllowed(cdpSrc) cdpAllowed(cdpDst) {
        require(ilks[cdpSrc] == ilks[cdpDst], "non-matching-cdps");
        (uint ink, uint art) = VatLike(vat).urns(ilks[cdpSrc], urns[cdpSrc]);
        VatLike(vat).fork(
            ilks[cdpSrc],
            urns[cdpSrc],
            urns[cdpDst],
            toInt(ink),
            toInt(art)
        );
    }
```

12.3 contract GetCdps

```
contract GetCdps {
}
```

12.3.1 getCdpsAsc(manager, guy)

```
    function getCdpsAsc(address manager, address guy) external view returns (
        ↪ uint[] memory ids, address[] memory urns, bytes32[] memory ilks) {
        uint count = DssCdpManager(manager).count(guy);
        ids = new uint[](count);
        urns = new address[](count);
        ilks = new bytes32[](count);
        uint i = 0;
        uint id = DssCdpManager(manager).first(guy);

        while (id > 0) {
            ids[i] = id;
            urns[i] = DssCdpManager(manager).urns(id);
            ilks[i] = DssCdpManager(manager).ilks(id);
            (,id) = DssCdpManager(manager).list(id);
            i++;
        }
    }
```

12.3.2 getCdpsDesc(manager, guy)

```
    function getCdpsDesc(address manager, address guy) external view returns (
        ↪ uint[] memory ids, address[] memory urns, bytes32[] memory ilks) {
        uint count = DssCdpManager(manager).count(guy);
        ids = new uint[](count);
        urns = new address[](count);
        ilks = new bytes32[](count);
        uint i = 0;
        uint id = DssCdpManager(manager).last(guy);

        while (id > 0) {
            ids[i] = id;
            urns[i] = DssCdpManager(manager).urns(id);
            ilks[i] = DssCdpManager(manager).ilks(id);
            (id,) = DssCdpManager(manager).list(id);
            i++;
        }
    }
```

12.4 contract DssProxyActions

```
contract DssProxyActions is Common {
}
```

Inherited:

```
//
    ↪ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    ↪
// WARNING: These functions meant to be used as a a library for a DSProxy. Some
    ↪ are unsafe if you call them directly.
//
    ↪ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    ↪

contract Common {
    uint256 constant RAY = 10 ** 27;
}
```

12.4.1 mul(x, y) [Common]

```
    // Internal functions

    function mul(uint x, uint y) internal pure returns (uint z) {
        require(y == 0 || (z = x * y) / y == x, "mul-overflow");
    }
```

12.4.2 daiJoin_join(apt, urn, wad) [Common] X

```
    // Public functions

    function daiJoin_join(address apt, address urn, uint wad) public {
        // Gets DAI from the user's wallet
        DaiJoinLike(apt).dai().transferFrom(msg.sender, address(this), wad);
        // Approves adapter to take the DAI amount
        DaiJoinLike(apt).dai().approve(apt, wad);
        // Joins DAI into the vat
        DaiJoinLike(apt).join(urn, wad);
    }
```

12.4.3 sub(x, y)

```
    // Internal functions

    function sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x, "sub-overflow");
    }
```

12.4.4 toInt(x)

```
    function toInt(uint x) internal pure returns (int y) {
        y = int(x);
        require(y >= 0, "int-overflow");
    }
```

12.4.5 toRad(wad)

```
    function toRad(uint wad) internal pure returns (uint rad) {
        rad = mul(wad, 10 ** 27);
    }
```

12.4.6 convertTo18(gemJoin, amt)

```
    function convertTo18(address gemJoin, uint256 amt) internal returns (uint256
        ↪  wad) {
        // For those collaterals that have less than 18 decimals precision we
            ↪ need to do the conversion before passing to frob function
        // Adapters will automatically handle the difference of precision
        wad = mul(
            amt,
            10 ** (18 - GemJoinLike(gemJoin).dec())
        );
    }
```

12.4.7 _getDrawDart(vat, jug, urn, ilk, wad)

```
    function _getDrawDart(
        address vat,
        address jug,
        address urn,
        bytes32 ilk,
        uint wad
    ) internal returns (int dart) {
        // Updates stability fee rate
        uint rate = JugLike(jug).drip(ilk);

        // Gets DAI balance of the urn in the vat
        uint dai = VatLike(vat).dai(urn);

        // If there was already enough DAI in the vat balance, just exits it
            ↪ without adding more debt
        if (dai < mul(wad, RAY)) {
            // Calculates the needed dart so together with the existing dai in
                ↪ the vat is enough to exit wad amount of DAI tokens
            dart = toInt(sub(mul(wad, RAY), dai) / rate);
            // This is neeeded due lack of precision. It might need to sum an
                ↪ extra dart wei (for the given DAI wad amount)
            dart = mul(uint(dart), rate) < mul(wad, RAY) ? dart + 1 : dart;
        }
    }
```

12.4.8 _getWipeDart(vat, dai, urn, ilk)

```
    function _getWipeDart(
        address vat,
        uint dai,
        address urn,
        bytes32 ilk
    ) internal view returns (int dart) {
        // Gets actual rate from the vat
        (, uint rate,,,) = VatLike(vat).ilks(ilk);
        // Gets actual art value of the urn
        (, uint art) = VatLike(vat).urns(ilk, urn);

        // Uses the whole dai balance in the vat to reduce the debt
        dart = toInt(dai / rate);
        // Checks the calculated dart is not higher than urn.art (total debt),
            ↪ otherwise uses its value
        dart = uint(dart) <= art ? - dart : - toInt(art);
    }
```

12.4.9 _getWipeAllWad(vat, usr, urn, ilk)

```
    function _getWipeAllWad(
        address vat,
        address usr,
        address urn,
        bytes32 ilk
    ) internal view returns (uint wad) {
        // Gets actual rate from the vat
        (, uint rate,,,) = VatLike(vat).ilks(ilk);
        // Gets actual art value of the urn
        (, uint art) = VatLike(vat).urns(ilk, urn);
        // Gets actual dai amount in the urn
        uint dai = VatLike(vat).dai(usr);

        uint rad = sub(mul(art, rate), dai);
        wad = rad / RAY;

        // If the rad precision has some dust, it will need to request for 1
            ↪ extra wad wei
        wad = mul(wad, RAY) < rad ? wad + 1 : wad;
    }
```

12.4.10 transfer(gem, dst, amt) X

```
    // Public functions

    function transfer(address gem, address dst, uint amt) public {
        GemLike(gem).transfer(dst, amt);
    }
```

12.4.11 ethJoin_join(apt, urn) X

```
    function ethJoin_join(address apt, address urn) public payable {
        // Wraps ETH in WETH
        GemJoinLike(apt).gem().deposit.value(msg.value)();
        // Approves adapter to take the WETH amount
        GemJoinLike(apt).gem().approve(address(apt), msg.value);
        // Joins WETH collateral into the vat
        GemJoinLike(apt).join(urn, msg.value);
    }
```

12.4.12 gemJoin_join(apt, urn, amt, transferFrom) X

```
    function gemJoin_join(address apt, address urn, uint amt, bool transferFrom)
        ↪  public {
        // Only executes for tokens that have approval/transferFrom
            ↪ implementation
        if (transferFrom) {
            // Gets token from the user's wallet
            GemJoinLike(apt).gem().transferFrom(msg.sender, address(this), amt);
            // Approves adapter to take the token amount
            GemJoinLike(apt).gem().approve(apt, amt);
        }
        // Joins token collateral into the vat
        GemJoinLike(apt).join(urn, amt);
    }
```

12.4.13 hope(obj, usr) X

```
    function hope(
        address obj,
        address usr
    ) public {
        HopeLike(obj).hope(usr);
    }
```

12.4.14 nope(obj, usr) X

```
function nope(
    address obj,
    address usr
) public {
    HopeLike(obj).nope(usr);
}
```

12.4.15 open(manager, ilk, usr) X

```
function open(
    address manager,
    bytes32 ilk,
    address usr
) public returns (uint cdp) {
    cdp = ManagerLike(manager).open(ilk, usr);
}
```

12.4.16 give(manager, cdp, usr) X

```
function give(
    address manager,
    uint cdp,
    address usr
) public {
    ManagerLike(manager).give(cdp, usr);
}
```

12.4.17 giveToProxy(proxyRegistry, manager, cdp, dst) X

```
function giveToProxy(
    address proxyRegistry,
    address manager,
    uint cdp,
    address dst
) public {
    // Gets actual proxy address
    address proxy = ProxyRegistryLike(proxyRegistry).proxies(dst);
    // Checks if the proxy address already existed and dst address is still
       ↪ the owner
    if (proxy == address(0) || ProxyLike(proxy).owner() != dst) {
        uint csize;
        assembly {
            csize := extcodesize(dst)
        }
        // We want to avoid creating a proxy for a contract address that
           ↪ might not be able to handle proxies, then losing the CDP
        require(csize == 0, "Dst-is-a-contract");
        // Creates the proxy for the dst address
        proxy = ProxyRegistryLike(proxyRegistry).build(dst);
    }
    // Transfers CDP to the dst proxy
    give(manager, cdp, proxy);
}
```

12.4.18 cdpAllow(manager, cdp, usr, ok) X

```
function cdpAllow(
    address manager,
    uint cdp,
    address usr,
    uint ok
) public {
    ManagerLike(manager).cdpAllow(cdp, usr, ok);
}
```

12.4.19 urnAllow(manager, usr, ok) X

```
    function urnAllow(
        address manager,
        address usr,
        uint ok
    ) public {
        ManagerLike(manager).urnAllow(usr, ok);
    }
```

12.4.20 flux(manager, cdp, dst, wad) X

```
    function flux(
        address manager,
        uint cdp,
        address dst,
        uint wad
    ) public {
        ManagerLike(manager).flux(cdp, dst, wad);
    }
```

12.4.21 move(manager, cdp, dst, rad) X

```
    function move(
        address manager,
        uint cdp,
        address dst,
        uint rad
    ) public {
        ManagerLike(manager).move(cdp, dst, rad);
    }
```

12.4.22 frob(manager, cdp, dink, dart) X

```
    function frob(
        address manager,
        uint cdp,
        int dink,
        int dart
    ) public {
        ManagerLike(manager).frob(cdp, dink, dart);
    }
```

12.4.23 quit(manager, cdp, dst) X

```
    function quit(
        address manager,
        uint cdp,
        address dst
    ) public {
        ManagerLike(manager).quit(cdp, dst);
    }
```

12.4.24 enter(manager, src, cdp) X

```
    function enter(
        address manager,
        address src,
        uint cdp
    ) public {
        ManagerLike(manager).enter(src, cdp);
    }
```

12.4.25 shift(manager, cdpSrc, cdpOrg) X

```
    function shift(
        address manager,
        uint cdpSrc,
        uint cdpOrg
    ) public {
        ManagerLike(manager).shift(cdpSrc, cdpOrg);
    }
```

12.4.26 makeGemBag(gemJoin) X

```
    function makeGemBag(
        address gemJoin
    ) public returns (address bag) {
        bag = GNTJoinLike(gemJoin).make(address(this));
    }
```

12.4.27 lockETH(manager, ethJoin, cdp) X

```
    function lockETH(
        address manager,
        address ethJoin,
        uint cdp
    ) public payable {
        // Receives ETH amount, converts it to WETH and joins it into the vat
        ethJoin_join(ethJoin, address(this));
        // Locks WETH amount into the CDP
        VatLike(ManagerLike(manager).vat()).frob(
            ManagerLike(manager).ilks(cdp),
            ManagerLike(manager).urns(cdp),
            address(this),
            address(this),
            toInt(msg.value),
            0
        );
    }
```

12.4.28 safeLockETH(manager, ethJoin, cdp, owner) X

```
    function safeLockETH(
        address manager,
        address ethJoin,
        uint cdp,
        address owner
    ) public payable {
        require(ManagerLike(manager).owns(cdp) == owner, "owner-missmatch");
        lockETH(manager, ethJoin, cdp);
    }
```

12.4.29 lockGem(manager, gemJoin, cdp, amt, transferFrom) X

```
    function lockGem(
        address manager,
        address gemJoin,
        uint cdp,
        uint amt,
        bool transferFrom
    ) public {
        // Takes token amount from user's wallet and joins into the vat
        gemJoin_join(gemJoin, address(this), amt, transferFrom);
        // Locks token amount into the CDP
        VatLike(ManagerLike(manager).vat()).frob(
            ManagerLike(manager).ilks(cdp),
            ManagerLike(manager).urns(cdp),
            address(this),
            address(this),
            toInt(convertTo18(gemJoin, amt)),
            0
        );
    }
```

12.4.30 safeLockGem(manager, gemJoin, cdp, amt, transferFrom, owner) X

```
    function safeLockGem(
        address manager,
        address gemJoin,
        uint cdp,
        uint amt,
        bool transferFrom,
        address owner
    ) public {
        require(ManagerLike(manager).owns(cdp) == owner, "owner-missmatch");
        lockGem(manager, gemJoin, cdp, amt, transferFrom);
    }
```

12.4.31 freeETH(manager, ethJoin, cdp, wad) X

```
    function freeETH(
        address manager,
        address ethJoin,
        uint cdp,
        uint wad
    ) public {
        // Unlocks WETH amount from the CDP
        frob(manager, cdp, -toInt(wad), 0);
        // Moves the amount from the CDP urn to proxy's address
        flux(manager, cdp, address(this), wad);
        // Exits WETH amount to proxy address as a token
        GemJoinLike(ethJoin).exit(address(this), wad);
        // Converts WETH to ETH
        GemJoinLike(ethJoin).gem().withdraw(wad);
        // Sends ETH back to the user's wallet
        msg.sender.transfer(wad);
    }
```

12.4.32 freeGem(manager, gemJoin, cdp, amt) X

```
    function freeGem(
        address manager,
        address gemJoin,
        uint cdp,
        uint amt
    ) public {
        uint wad = convertTo18(gemJoin, amt);
        // Unlocks token amount from the CDP
        frob(manager, cdp, -toInt(wad), 0);
        // Moves the amount from the CDP urn to proxy's address
        flux(manager, cdp, address(this), wad);
        // Exits token amount to the user's wallet as a token
        GemJoinLike(gemJoin).exit(msg.sender, amt);
    }
```

12.4.33 exitETH(manager, ethJoin, cdp, wad) X

```
    function exitETH(
        address manager,
        address ethJoin,
        uint cdp,
        uint wad
    ) public {
        // Moves the amount from the CDP urn to proxy's address
        flux(manager, cdp, address(this), wad);

        // Exits WETH amount to proxy address as a token
        GemJoinLike(ethJoin).exit(address(this), wad);
        // Converts WETH to ETH
        GemJoinLike(ethJoin).gem().withdraw(wad);
        // Sends ETH back to the user's wallet
        msg.sender.transfer(wad);
    }
```

12.4.34 exitGem(manager, gemJoin, cdp, amt) X

```
    function exitGem(
        address manager,
        address gemJoin,
        uint cdp,
        uint amt
    ) public {
        // Moves the amount from the CDP urn to proxy's address
        flux(manager, cdp, address(this), convertTo18(gemJoin, amt));

        // Exits token amount to the user's wallet as a token
        GemJoinLike(gemJoin).exit(msg.sender, amt);
    }
```

12.4.35 draw(manager, jug, daiJoin, cdp, wad) X

```
    function draw(
        address manager,
        address jug,
        address daiJoin,
        uint cdp,
        uint wad
    ) public {
        address urn = ManagerLike(manager).urns(cdp);
        address vat = ManagerLike(manager).vat();
        bytes32 ilk = ManagerLike(manager).ilks(cdp);
        // Generates debt in the CDP
        frob(manager, cdp, 0, _getDrawDart(vat, jug, urn, ilk, wad));
        // Moves the DAI amount (balance in the vat in rad) to proxy's address
        move(manager, cdp, address(this), toRad(wad));
        // Allows adapter to access to proxy's DAI balance in the vat
        if (VatLike(vat).can(address(this), address(daiJoin)) == 0) {
            VatLike(vat).hope(daiJoin);
        }
        // Exits DAI to the user's wallet as a token
        DaiJoinLike(daiJoin).exit(msg.sender, wad);
    }
```

12.4.36 wipe(manager, daiJoin, cdp, wad) X

```
    function wipe(
        address manager,
        address daiJoin,
        uint cdp,
        uint wad
    ) public {
        address vat = ManagerLike(manager).vat();
        address urn = ManagerLike(manager).urns(cdp);
        bytes32 ilk = ManagerLike(manager).ilks(cdp);

        address own = ManagerLike(manager).owns(cdp);
        if (own == address(this) || ManagerLike(manager).cdpCan(own, cdp,
            ↪ address(this)) == 1) {
            // Joins DAI amount into the vat
            daiJoin_join(daiJoin, urn, wad);
            // Paybacks debt to the CDP
            frob(manager, cdp, 0, _getWipeDart(vat, VatLike(vat).dai(urn), urn,
                ↪ ilk));
        } else {
             // Joins DAI amount into the vat
            daiJoin_join(daiJoin, address(this), wad);
            // Paybacks debt to the CDP
            VatLike(vat).frob(
                ilk,
                urn,
                address(this),
                address(this),
                0,
                _getWipeDart(vat, wad * RAY, urn, ilk)
            );
        }
    }
```

12.4.37 safeWipe(manager, daiJoin, cdp, wad, owner) X

```
    function safeWipe(
        address manager,
        address daiJoin,
        uint cdp,
        uint wad,
        address owner
    ) public {
        require(ManagerLike(manager).owns(cdp) == owner, "owner-missmatch");
        wipe(manager, daiJoin, cdp, wad);
    }
```

12.4.38 wipeAll(manager, daiJoin, cdp) X

```
    function wipeAll(
        address manager,
        address daiJoin,
        uint cdp
    ) public {
        address vat = ManagerLike(manager).vat();
        address urn = ManagerLike(manager).urns(cdp);
        bytes32 ilk = ManagerLike(manager).ilks(cdp);
        (, uint art) = VatLike(vat).urns(ilk, urn);

        address own = ManagerLike(manager).owns(cdp);
        if (own == address(this) || ManagerLike(manager).cdpCan(own, cdp,
            ↪ address(this)) == 1) {
            // Joins DAI amount into the vat
            daiJoin_join(daiJoin, urn, _getWipeAllWad(vat, urn, urn, ilk));
            // Paybacks debt to the CDP
            frob(manager, cdp, 0, -int(art));
        } else {
            // Joins DAI amount into the vat
            daiJoin_join(daiJoin, address(this), _getWipeAllWad(vat, address(
                ↪ this), urn, ilk));
            // Paybacks debt to the CDP
            VatLike(vat).frob(
                ilk,
                urn,
                address(this),
                address(this),
                0,
                -int(art)
            );
        }
    }
```

12.4.39 safeWipeAll(manager, daiJoin, cdp, owner) X

```
    function safeWipeAll(
        address manager,
        address daiJoin,
        uint cdp,
        address owner
    ) public {
        require(ManagerLike(manager).owns(cdp) == owner, "owner-missmatch");
        wipeAll(manager, daiJoin, cdp);
    }
```

12.4.40 lockETHAndDraw(manager, jug, ethJoin, daiJoin, cdp, wadD) X

```
    function lockETHAndDraw(
        address manager,
        address jug,
        address ethJoin,
        address daiJoin,
        uint cdp,
        uint wadD
    ) public payable {
        address urn = ManagerLike(manager).urns(cdp);
        address vat = ManagerLike(manager).vat();
        bytes32 ilk = ManagerLike(manager).ilks(cdp);
        // Receives ETH amount, converts it to WETH and joins it into the vat
        ethJoin_join(ethJoin, urn);
        // Locks WETH amount into the CDP and generates debt
        frob(manager, cdp, toInt(msg.value), _getDrawDart(vat, jug, urn, ilk,
            ↪ wadD));
        // Moves the DAI amount (balance in the vat in rad) to proxy's address
        move(manager, cdp, address(this), toRad(wadD));
        // Allows adapter to access to proxy's DAI balance in the vat
        if (VatLike(vat).can(address(this), address(daiJoin)) == 0) {
            VatLike(vat).hope(daiJoin);
        }
        // Exits DAI to the user's wallet as a token
        DaiJoinLike(daiJoin).exit(msg.sender, wadD);
    }
```

12.4.41 openLockETHAndDraw(manager, jug, ethJoin, daiJoin, ilk, wadD) X

```
    function openLockETHAndDraw(
        address manager,
        address jug,
        address ethJoin,
        address daiJoin,
        bytes32 ilk,
        uint wadD
    ) public payable returns (uint cdp) {
        cdp = open(manager, ilk, address(this));
        lockETHAndDraw(manager, jug, ethJoin, daiJoin, cdp, wadD);
    }
```

12.4.42 lockGemAndDraw(manager, jug, gemJoin, daiJoin, cdp, amtC, wadD, transferFrom) X

```
    function lockGemAndDraw(
        address manager,
        address jug,
        address gemJoin,
        address daiJoin,
        uint cdp,
        uint amtC,
        uint wadD,
        bool transferFrom
    ) public {
        address urn = ManagerLike(manager).urns(cdp);
        address vat = ManagerLike(manager).vat();
        bytes32 ilk = ManagerLike(manager).ilks(cdp);
        // Takes token amount from user's wallet and joins into the vat
        gemJoin_join(gemJoin, urn, amtC, transferFrom);
        // Locks token amount into the CDP and generates debt
        frob(manager, cdp, toInt(convertTo18(gemJoin, amtC)), _getDrawDart(vat,
            ↪ jug, urn, ilk, wadD));
        // Moves the DAI amount (balance in the vat in rad) to proxy's address
        move(manager, cdp, address(this), toRad(wadD));
        // Allows adapter to access to proxy's DAI balance in the vat
        if (VatLike(vat).can(address(this), address(daiJoin)) == 0) {
            VatLike(vat).hope(daiJoin);
        }
        // Exits DAI to the user's wallet as a token
        DaiJoinLike(daiJoin).exit(msg.sender, wadD);
    }
```

12.4.43 openLockGemAndDraw(manager, jug, gemJoin, daiJoin, ilk, amtC, wadD, transferFrom) X

```
    function openLockGemAndDraw(
        address manager,
        address jug,
        address gemJoin,
        address daiJoin,
        bytes32 ilk,
        uint amtC,
        uint wadD,
        bool transferFrom
    ) public returns (uint cdp) {
        cdp = open(manager, ilk, address(this));
        lockGemAndDraw(manager, jug, gemJoin, daiJoin, cdp, amtC, wadD,
            ↪ transferFrom);
    }
```

12.4.44 openLockGNTAndDraw(manager, jug, gntJoin, daiJoin, ilk, amtC, wadD) X

```
    function openLockGNTAndDraw(
        address manager,
        address jug,
        address gntJoin,
        address daiJoin,
        bytes32 ilk,
        uint amtC,
        uint wadD
    ) public returns (address bag, uint cdp) {
        // Creates bag (if doesn't exist) to hold GNT
        bag = GNTJoinLike(gntJoin).bags(address(this));
        if (bag == address(0)) {
            bag = makeGemBag(gntJoin);
        }
        // Transfer funds to the funds which previously were sent to the proxy
        GemLike(GemJoinLike(gntJoin).gem()).transfer(bag, amtC);
        cdp = openLockGemAndDraw(manager, jug, gntJoin, daiJoin, ilk, amtC, wadD
            ↪ , false);
    }
```

12.4.45 wipeAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC, wadD) X

```
    function wipeAndFreeETH(
        address manager,
        address ethJoin,
        address daiJoin,
        uint cdp,
        uint wadC,
        uint wadD
    ) public {
        address urn = ManagerLike(manager).urns(cdp);
        // Joins DAI amount into the vat
        daiJoin_join(daiJoin, urn, wadD);
        // Paybacks debt to the CDP and unlocks WETH amount from it
        frob(
            manager,
            cdp,
            -toInt(wadC),
            _getWipeDart(ManagerLike(manager).vat(), VatLike(ManagerLike(manager
                ↪ ).vat()).dai(urn), urn, ManagerLike(manager).ilks(cdp))
        );
        // Moves the amount from the CDP urn to proxy's address
        flux(manager, cdp, address(this), wadC);
        // Exits WETH amount to proxy address as a token
        GemJoinLike(ethJoin).exit(address(this), wadC);
        // Converts WETH to ETH
        GemJoinLike(ethJoin).gem().withdraw(wadC);
        // Sends ETH back to the user's wallet
        msg.sender.transfer(wadC);
    }
```

12.4.46 wipeAllAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC) X

```
    function wipeAllAndFreeETH(
        address manager,
        address ethJoin,
        address daiJoin,
        uint cdp,
        uint wadC
    ) public {
        address vat = ManagerLike(manager).vat();
        address urn = ManagerLike(manager).urns(cdp);
        bytes32 ilk = ManagerLike(manager).ilks(cdp);
        (, uint art) = VatLike(vat).urns(ilk, urn);
        // Joins DAI amount into the vat
        daiJoin_join(daiJoin, urn, _getWipeAllWad(vat, urn, urn, ilk));
        // Paybacks debt to the CDP and unlocks WETH amount from it
        frob(
            manager,
            cdp,
            -toInt(wadC),
            -int(art)
        );
        // Moves the amount from the CDP urn to proxy's address
        flux(manager, cdp, address(this), wadC);
        // Exits WETH amount to proxy address as a token
        GemJoinLike(ethJoin).exit(address(this), wadC);
        // Converts WETH to ETH
        GemJoinLike(ethJoin).gem().withdraw(wadC);
        // Sends ETH back to the user's wallet
        msg.sender.transfer(wadC);
    }
```

12.4.47 wipeAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC, wadD) X

```
    function wipeAndFreeGem(
        address manager,
        address gemJoin,
        address daiJoin,
        uint cdp,
        uint amtC,
        uint wadD
    ) public {
        address urn = ManagerLike(manager).urns(cdp);
        // Joins DAI amount into the vat
        daiJoin_join(daiJoin, urn, wadD);
        uint wadC = convertTo18(gemJoin, amtC);
        // Paybacks debt to the CDP and unlocks token amount from it
        frob(
            manager,
            cdp,
            -toInt(wadC),
            _getWipeDart(ManagerLike(manager).vat(), VatLike(ManagerLike(manager
                ↪ ).vat()).dai(urn), urn, ManagerLike(manager).ilks(cdp))
        );
        // Moves the amount from the CDP urn to proxy's address
        flux(manager, cdp, address(this), wadC);
        // Exits token amount to the user's wallet as a token
        GemJoinLike(gemJoin).exit(msg.sender, amtC);
    }
```

12.4.48 wipeAllAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC) X

```
    function wipeAllAndFreeGem(
        address manager,
        address gemJoin,
        address daiJoin,
        uint cdp,
        uint amtC
    ) public {
        address vat = ManagerLike(manager).vat();
        address urn = ManagerLike(manager).urns(cdp);
        bytes32 ilk = ManagerLike(manager).ilks(cdp);
        (, uint art) = VatLike(vat).urns(ilk, urn);

        // Joins DAI amount into the vat
        daiJoin_join(daiJoin, urn, _getWipeAllWad(vat, urn, urn, ilk));
        uint wadC = convertTo18(gemJoin, amtC);
        // Paybacks debt to the CDP and unlocks token amount from it
        frob(
            manager,
            cdp,
            -toInt(wadC),
            -int(art)
        );
        // Moves the amount from the CDP urn to proxy's address
        flux(manager, cdp, address(this), wadC);
        // Exits token amount to the user's wallet as a token
        GemJoinLike(gemJoin).exit(msg.sender, amtC);
    }
```

12.5 contract DssProxyActionsEnd

```
contract DssProxyActionsEnd is Common {
}
```

Inherited:

```
//
    ↪ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    ↪
// WARNING: These functions meant to be used as a a library for a DSProxy. Some
    ↪ are unsafe if you call them directly.
//
    ↪ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    ↪

contract Common {
    uint256 constant RAY = 10 ** 27;
}
```

12.5.1 mul(x, y) [Common]

```
    // Internal functions

    function mul(uint x, uint y) internal pure returns (uint z) {
        require(y == 0 || (z = x * y) / y == x, "mul-overflow");
    }
```

12.5.2 daiJoin_join(apt, urn, wad) [Common] X

```
    // Public functions

    function daiJoin_join(address apt, address urn, uint wad) public {
        // Gets DAI from the user's wallet
        DaiJoinLike(apt).dai().transferFrom(msg.sender, address(this), wad);
        // Approves adapter to take the DAI amount
        DaiJoinLike(apt).dai().approve(apt, wad);
        // Joins DAI into the vat
        DaiJoinLike(apt).join(urn, wad);
    }
```

12.5.3 _free(manager, end, cdp)

```
    // Internal functions

    function _free(
        address manager,
        address end,
        uint cdp
    ) internal returns (uint ink) {
        bytes32 ilk = ManagerLike(manager).ilks(cdp);
        address urn = ManagerLike(manager).urns(cdp);
        VatLike vat = VatLike(ManagerLike(manager).vat());
        uint art;
        (ink, art) = vat.urns(ilk, urn);

        // If CDP still has debt, it needs to be paid
        if (art > 0) {
            EndLike(end).skim(ilk, urn);
            (ink,) = vat.urns(ilk, urn);
        }
        // Approves the manager to transfer the position to proxy's address in
            ↪ the vat
        if (vat.can(address(this), address(manager)) == 0) {
            vat.hope(manager);
        }
        // Transfers position from CDP to the proxy address
        ManagerLike(manager).quit(cdp, address(this));
        // Frees the position and recovers the collateral in the vat registry
        EndLike(end).free(ilk);
    }
```

12.5.4 freeETH(manager, ethJoin, end, cdp) X

```
    // Public functions
    function freeETH(
        address manager,
        address ethJoin,
        address end,
        uint cdp
    ) public {
        uint wad = _free(manager, end, cdp);
        // Exits WETH amount to proxy address as a token
        GemJoinLike(ethJoin).exit(address(this), wad);
        // Converts WETH to ETH
        GemJoinLike(ethJoin).gem().withdraw(wad);
        // Sends ETH back to the user's wallet
        msg.sender.transfer(wad);
    }
```

12.5.5 freeGem(manager, gemJoin, end, cdp) X

```
    function freeGem(
        address manager,
        address gemJoin,
        address end,
        uint cdp
    ) public {
        uint amt = _free(manager, end, cdp) / 10 ** (18 - GemJoinLike(gemJoin).
            ↪ dec());
        // Exits token amount to the user's wallet as a token
        GemJoinLike(gemJoin).exit(msg.sender, amt);
    }
```

12.5.6 pack(daiJoin, end, wad) X

```
    function pack(
        address daiJoin,
        address end,
        uint wad
    ) public {
        daiJoin_join(daiJoin, address(this), wad);
        VatLike vat = DaiJoinLike(daiJoin).vat();
        // Approves the end to take out DAI from the proxy's balance in the vat
        if (vat.can(address(this), address(end)) == 0) {
            vat.hope(end);
        }
        EndLike(end).pack(wad);
    }
```

12.5.7 cashETH(ethJoin, end, ilk, wad) X

```
    function cashETH(
        address ethJoin,
        address end,
        bytes32 ilk,
        uint wad
    ) public {
        EndLike(end).cash(ilk, wad);
        uint wadC = mul(wad, EndLike(end).fix(ilk)) / RAY;
        // Exits WETH amount to proxy address as a token
        GemJoinLike(ethJoin).exit(address(this), wadC);
        // Converts WETH to ETH
        GemJoinLike(ethJoin).gem().withdraw(wadC);
        // Sends ETH back to the user's wallet
        msg.sender.transfer(wadC);
    }
```

12.5.8 cashGem(gemJoin, end, ilk, wad) X

```
    function cashGem(
        address gemJoin,
        address end,
        bytes32 ilk,
        uint wad
    ) public {
        EndLike(end).cash(ilk, wad);
        // Exits token amount to the user's wallet as a token
        uint amt = mul(wad, EndLike(end).fix(ilk)) / RAY / 10 ** (18 -
            ↪ GemJoinLike(gemJoin).dec());
        GemJoinLike(gemJoin).exit(msg.sender, amt);
    }
```

12.6 contract DssProxyActionsDsr

```
contract DssProxyActionsDsr is Common {
}
```

Inherited:

```
//
    ↪ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    ↪
// WARNING: These functions meant to be used as a a library for a DSProxy. Some
    ↪ are unsafe if you call them directly.
//
    ↪ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    ↪

contract Common {
    uint256 constant RAY = 10 ** 27;
}
```

12.6.1 mul(x, y) [Common]

```
    // Internal functions

    function mul(uint x, uint y) internal pure returns (uint z) {
        require(y == 0 || (z = x * y) / y == x, "mul-overflow");
    }
```

12.6.2 daiJoin_join(apt, urn, wad) [Common] X

```
    // Public functions

    function daiJoin_join(address apt, address urn, uint wad) public {
        // Gets DAI from the user's wallet
        DaiJoinLike(apt).dai().transferFrom(msg.sender, address(this), wad);
        // Approves adapter to take the DAI amount
        DaiJoinLike(apt).dai().approve(apt, wad);
        // Joins DAI into the vat
        DaiJoinLike(apt).join(urn, wad);
    }
```

12.6.3 join(daiJoin, pot, wad) X

```
    function join(
        address daiJoin,
        address pot,
        uint wad
    ) public {
        VatLike vat = DaiJoinLike(daiJoin).vat();
        // Executes drip to get the chi rate updated to rho == now, otherwise
            ↪ join will fail
        uint chi = PotLike(pot).drip();
        // Joins wad amount to the vat balance
        daiJoin_join(daiJoin, address(this), wad);
        // Approves the pot to take out DAI from the proxy's balance in the vat
        if (vat.can(address(this), address(pot)) == 0) {
            vat.hope(pot);
        }
        // Joins the pie value (equivalent to the DAI wad amount) in the pot
        PotLike(pot).join(mul(wad, RAY) / chi);
    }
```

12.6.4 exit(daiJoin, pot, wad) X

```
    function exit(
        address daiJoin,
        address pot,
        uint wad
    ) public {
        VatLike vat = DaiJoinLike(daiJoin).vat();
        // Executes drip to count the savings accumulated until this moment
        uint chi = PotLike(pot).drip();
        // Calculates the pie value in the pot equivalent to the DAI wad amount
        uint pie = mul(wad, RAY) / chi;
        // Exits DAI from the pot
        PotLike(pot).exit(pie);
        // Checks the actual balance of DAI in the vat after the pot exit
        uint bal = DaiJoinLike(daiJoin).vat().dai(address(this));
        // Allows adapter to access to proxy's DAI balance in the vat
        if (vat.can(address(this), address(daiJoin)) == 0) {
            vat.hope(daiJoin);
        }
        // It is necessary to check if due rounding the exact wad amount can be
            ↪ exited by the adapter.
        // Otherwise it will do the maximum DAI balance in the vat
        DaiJoinLike(daiJoin).exit(
            msg.sender,
            bal >= mul(wad, RAY) ? wad : bal / RAY
        );
    }
```

12.6.5 exitAll(daiJoin, pot) X

```
    function exitAll(
        address daiJoin,
        address pot
    ) public {
        VatLike vat = DaiJoinLike(daiJoin).vat();
        // Executes drip to count the savings accumulated until this moment
        uint chi = PotLike(pot).drip();
        // Gets the total pie belonging to the proxy address
        uint pie = PotLike(pot).pie(address(this));
        // Exits DAI from the pot
        PotLike(pot).exit(pie);
        // Allows adapter to access to proxy's DAI balance in the vat
        if (vat.can(address(this), address(daiJoin)) == 0) {
            vat.hope(daiJoin);
        }
        // Exits the DAI amount corresponding to the value of pie
        DaiJoinLike(daiJoin).exit(msg.sender, mul(chi, pie) / RAY);
    }
```

Chapter 13

Exchange callees

13.1 contract CalleeMakerOtcDai

```
// Maker-Otc is MatchingMarket, which is the core contract of OasisDex
contract CalleeMakerOtcDai is CalleeMakerOtc {
}
```

Inherited:

```
// Simple Callee Example to interact with MatchingMarket
// This Callee contract exists as a standalone contract
contract CalleeMakerOtc {
    OtcLike          public otc;
    DaiJoinLike      public daiJoin;
    TokenLike        public dai;

    uint256          public constant RAY = 10 ** 27;
}
```

13.1.1 add(x, y) [CalleeMakerOtc]

```
    function add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

13.1.2 sub(x, y) [CalleeMakerOtc]

```
    function sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

13.1.3 divup(x, y) [CalleeMakerOtc]

```
    function divup(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = add(x, sub(y, 1)) / y;
    }
```

13.1.4 setUp(otc_, daiJoin_) [CalleeMakerOtc]

```
    function setUp(address otc_, address daiJoin_) internal {
        otc = OtcLike(otc_);
        daiJoin = DaiJoinLike(daiJoin_);
        dai = daiJoin.dai();

        dai.approve(daiJoin_, uint256(-1));
    }
```

13.1.5 _fromWad(gemJoin, wad) [CalleeMakerOtc]

```
    function _fromWad(address gemJoin, uint256 wad) internal view returns (
        ↪ uint256 amt) {
        amt = wad / 10 ** (sub(18, GemJoinLike(gemJoin).dec()));
    }
```

13.1.6 constructor(otc_, daiJoin_) X

```
    constructor(address otc_, address daiJoin_) public {
        setUp(otc_, daiJoin_);
    }
```

13.1.7 clipperCall(sender, daiAmt, gemAmt, data) X

```
    function clipperCall(
        address sender,         // Clipper Caller and Dai deliveryaddress
        uint256 daiAmt,         // Dai amount to payback[rad]
        uint256 gemAmt,         // Gem amount received [wad]
        bytes calldata data     // Extra data needed (gemJoin)
) external {
        // Get address to send remaining DAI, gemJoin adapter and minProfit in
            ↪ DAI to make
        (
            address to,
            address gemJoin,
            uint256 minProfit,
            address charterManager
        ) = abi.decode(data, (address, address, uint256, address));

        // Convert gem amount to token precision
        gemAmt = _fromWad(gemJoin, gemAmt);

        // Exit collateral to token version
        if(charterManager != address(0)) {
            CharterManagerLike(charterManager).exit(gemJoin, address(this),
                ↪ gemAmt);
        } else {
            GemJoinLike(gemJoin).exit(address(this), gemAmt);
        }

        // Approve otc to take gem
        TokenLike gem = GemJoinLike(gemJoin).gem();
        gem.approve(address(otc), gemAmt);

        // Calculate amount of DAI to Join (as erc20 WAD value)
        uint256 daiToJoin = divup(daiAmt, RAY);

        // Do operation
        otc.sellAllAmount(address(gem), gemAmt, address(dai), add(daiToJoin,
            ↪ minProfit));

        // Although maker-otc reverts if order book is empty, this check is a
            ↪ sanity check for other exchanges
        // Transfer any lingering gem to specified address
        if (gem.balanceOf(address(this)) > 0) {
            gem.transfer(to, gem.balanceOf(address(this)));
        }

        // Convert DAI bought to internal vat value of the msg.sneder of Clipper
            ↪ .take
        daiJoin.join(sender, daiToJoin);

        // Transfer remaining DAI to specified address
        dai.transfer(to, dai.balanceOf(address(this)));
    }
```

13.2 contract CurveLpTokenUniv3Callee

```
contract CurveLpTokenUniv3Callee {
    UniV3RouterLike public immutable uniV3Router;
    DaiJoinLike     public immutable daiJoin;
    TokenLike       public immutable dai;
    address         public immutable weth;

    uint256         public constant RAY = 10 ** 27;
    address         public constant ETH = 0
        ↪ xEeeeeEeeeEeEeeEeEeeEEEeeeeEeeeeeeeEEeE;

    receive() external payable {}
}
```

13.2.1 struct CurveLpTokenUniv3Callee.CurveData

```
    struct CurveData {
        address pool;
        uint256 coinIndex;
    }
```

13.2.2 _add(x, y)

```
    function _add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

13.2.3 _sub(x, y)

```
    function _sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

13.2.4 _divup(x, y)

```
    function _divup(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = _add(x, _sub(y, 1)) / y;
    }
```

13.2.5 constructor(uniV3Router_, daiJoin_, weth_) X

```
    constructor(
        address uniV3Router_,
        address daiJoin_,
        address weth_
    ) public {
        uniV3Router     = UniV3RouterLike(uniV3Router_);
        daiJoin         = DaiJoinLike(daiJoin_);
        TokenLike dai_  = DaiJoinLike(daiJoin_).dai();
        dai             = dai_;
        weth            = weth_;

        dai_.approve(daiJoin_, type(uint256).max);
    }
```

13.2.6 _fromWad(gemJoin, wad)

```
    function _fromWad(address gemJoin, uint256 wad) internal view returns (
        ↪ uint256 amt) {
        amt = wad / 10 ** (_sub(18, GemJoinLike(gemJoin).dec()));
    }
```

13.2.7 clipperCall(sender, owe, slice, data) X

```
    function clipperCall(
        address sender,             // Clipper caller, pays back the loan
        uint256 owe,                // Dai amount to pay back         [rad]
        uint256 slice,              // Gem amount received            [wad]
        bytes calldata data         // Extra data, see below
    ) external {
        (
            address           to,         // address to send remaining DAI to
            address           gemJoin,    // gemJoin adapter address
            uint256           minProfit,  // minimum profit in DAI to make [wad]
            bytes memory      path,       // uniswap v3 path
            address           manager,    // pass address(0) if no manager
            CurveData memory  curveData   // curve pool data
        ) = abi.decode(data, (address, address, uint256, bytes, address,
            ↪ CurveData));

        address gem = GemJoinLike(gemJoin).gem();

        // Convert slice to token precision
        slice = _fromWad(gemJoin, slice);

        // Exit gem to token
        if(manager != address(0)) {
            ManagerLike(manager).exit(gemJoin, address(this), slice);
        } else {
            GemJoinLike(gemJoin).exit(address(this), slice);
        }

        // curveData used explicitly to avoid stack too deep
        TokenLike(gem).approve(curveData.pool, slice);
        slice = CurvePoolLike(curveData.pool).remove_liquidity_one_coin({
            _token_amount: slice,
            i:             int128(curveData.coinIndex),
            _min_amount:   0 // minProfit is checked below
        });

        gem = CurvePoolLike(curveData.pool).coins(curveData.coinIndex);
        if (gem == ETH) {
            gem = weth;
            WethLike(gem).deposit{
                value: slice
            }();
        }

        // Approve uniV3 to take gem
        TokenLike(gem).approve(address(uniV3Router), slice);

        // Calculate amount of DAI to Join (as erc20 WAD value)
        uint256 daiToJoin = _divup(owe, RAY);

        UniV3RouterLike.ExactInputParams memory params = UniV3RouterLike.
            ↪ ExactInputParams({
            path:             path,
            recipient:        address(this),
            deadline:         block.timestamp,
            amountIn:         slice,
            amountOutMinimum: _add(daiToJoin, minProfit)
        });
        uniV3Router.exactInput(params);

        // Although Uniswap will accept all gems, this check is a sanity check,
            ↪ just in case
        // Transfer any lingering gem to specified address
        if (TokenLike(gem).balanceOf(address(this)) > 0) {
            TokenLike(gem).transfer(to, TokenLike(gem).balanceOf(address(this)))
                ↪ ;
        }

        // Convert DAI bought to internal vat value of the msg.sender of Clipper
            ↪ .take
        daiJoin.join(sender, daiToJoin);

        // Transfer remaining DAI to specified address
        dai.transfer(to, dai.balanceOf(address(this)));
    }
```

13.3 contract PSMCallee

```
contract PSMCallee {
    DaiJoinLike              public daiJoin;
    TokenLike                public dai;

    uint256                  public constant RAY = 10 ** 27;
}
```

13.3.1 _add(x, y)

```
    function _add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

13.3.2 _sub(x, y)

```
    function _sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

13.3.3 _divup(x, y)

```
    function _divup(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = _add(x, _sub(y, 1)) / y;
    }
```

13.3.4 constructor(daiJoin_) X

```
    constructor(address daiJoin_) public {
        daiJoin = DaiJoinLike(daiJoin_);
        dai = daiJoin.dai();

        dai.approve(daiJoin_, uint256(-1));
    }
```

13.3.5 _fromWad(gemJoin, wad)

```
    function _fromWad(address gemJoin, uint256 wad) internal view returns (
        ↪ uint256 amt) {
        amt = wad / 10 ** (_sub(18, GemJoinLike(gemJoin).dec()));
    }
```

13.3.6 clipperCall(sender, owe, slice, data) X

```
    function clipperCall(
        address sender,            // Clipper caller, pays back the loan
        uint256 owe,               // Dai amount to pay back        [rad]
        uint256 slice,             // Gem amount received           [wad]
        bytes calldata data        // Extra data, see below
    ) external {
        (
            address to,            // address to send remaining DAI to
            address gemJoin,       // gemJoin adapter address
            uint256 minProfit,     // minimum profit in DAI to make [wad]
            address psm            // psm address for swapping collateral to DAI
        ) = abi.decode(data, (address, address, uint256, address));

        // Convert slice to token precision
        slice = _fromWad(gemJoin, slice);

        // Exit gem to token
        GemJoinLike(gemJoin).exit(address(this), slice);

        // Approve psm's gemJoin to take gem
        TokenLike gem = GemJoinLike(gemJoin).gem();
        gem.approve(PSMLike(psm).gemJoin(), slice);

        // Calculate amount of DAI to Join (as erc20 WAD value)
        uint256 daiToJoin = _divup(owe, RAY);

        PSMLike(psm).sellGem(address(this), slice);
        require(dai.balanceOf(address(this)) > _add(daiToJoin, minProfit), "Not
            ↪ enough dai from psm");

        // Although psm will accept all gems, this check is a sanity check, just
            ↪  in case
        // Transfer any lingering gem to specified address
        if (gem.balanceOf(address(this)) > 0) {
            gem.transfer(to, gem.balanceOf(address(this)));
        }

        // Convert DAI bought to internal vat value of the msg.sender of Clipper
            ↪ .take
        daiJoin.join(sender, daiToJoin);

        // Transfer remaining DAI to specified address
        dai.transfer(to, dai.balanceOf(address(this)));
    }
```

13.4 contract UniswapV2CalleeDai

```
// Uniswapv2Router02 route directs swaps from one pool to another
contract UniswapV2CalleeDai is UniswapV2Callee {
}
```

Inherited:

```
// Simple Callee Example to interact with MatchingMarket
// This Callee contract exists as a standalone contract
contract UniswapV2Callee {
    UniswapV2Router02Like   public uniRouter02;
    DaiJoinLike             public daiJoin;
    TokenLike               public dai;

    uint256                 public constant RAY = 10 ** 27;
}
```

13.4.1 add(x, y) [UniswapV2Callee]

```
    function add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

13.4.2 sub(x, y) [UniswapV2Callee]

```
    function sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

13.4.3 divup(x, y) [UniswapV2Callee]

```
    function divup(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = add(x, sub(y, 1)) / y;
    }
```

13.4.4 setUp(uniRouter02_, daiJoin_) [UniswapV2Callee]

```
    function setUp(address uniRouter02_, address daiJoin_) internal {
        uniRouter02 = UniswapV2Router02Like(uniRouter02_);
        daiJoin = DaiJoinLike(daiJoin_);
        dai = daiJoin.dai();

        dai.approve(daiJoin_, uint256(-1));
    }
```

13.4.5 _fromWad(gemJoin, wad) [UniswapV2Callee]

```
    function _fromWad(address gemJoin, uint256 wad) internal view returns (
       ↪ uint256 amt) {
        amt = wad / 10 ** (sub(18, GemJoinLike(gemJoin).dec()));
    }
```

13.4.6 constructor(uniRouter02_, daiJoin_) X

```
    constructor(address uniRouter02_, address daiJoin_) public {
        setUp(uniRouter02_, daiJoin_);
    }
```

13.4.7 clipperCall(sender, daiAmt, gemAmt, data) X

```
    function clipperCall(
        address sender,         // Clipper Caller and Dai deliveryaddress
        uint256 daiAmt,         // Dai amount to payback[rad]
        uint256 gemAmt,         // Gem amount received [wad]
        bytes calldata data     // Extra data needed (gemJoin)
    ) external {
        (
            address to,             // address to send remaining DAI to
            address gemJoin,        // gemJoin adapter address
            uint256 minProfit,      // minimum profit in DAI to make [wad]
            address[] memory path, // Uniswap pool path
            address charterManager // pass address(0) if no manager
        ) = abi.decode(data, (address, address, uint256, address[], address));

        // Convert gem amount to token precision
        gemAmt = _fromWad(gemJoin, gemAmt);

        // Exit collateral to token version
        if(charterManager != address(0)) {
            CharterManagerLike(charterManager).exit(gemJoin, address(this),
                ↪ gemAmt);
        } else {
            GemJoinLike(gemJoin).exit(address(this), gemAmt);
        }

        // Approve uniRouter02 to take gem
        TokenLike gem = GemJoinLike(gemJoin).gem();
        gem.approve(address(uniRouter02), gemAmt);

        // Calculate amount of DAI to Join (as erc20 WAD value)
        uint256 daiToJoin = divup(daiAmt, RAY);

        // Do operation and get dai amount bought (checking the profit is
            ↪ achieved)
        uniRouter02.swapExactTokensForTokens(
            gemAmt,
            add(daiToJoin, minProfit),
            path,
            address(this),
            block.timestamp
        );

        // Although Uniswap will accept all gems, this check is a sanity check,
            ↪ just in case
        // Transfer any lingering gem to specified address
        if (gem.balanceOf(address(this)) > 0) {
            gem.transfer(to, gem.balanceOf(address(this)));
        }

        // Convert DAI bought to internal vat value of the msg.sender of Clipper
            ↪ .take
        daiJoin.join(sender, daiToJoin);

        // Transfer remaining DAI to specified address
        dai.transfer(to, dai.balanceOf(address(this)));
    }
```

13.5 contract UniswapV2LpTokenCalleeDai

```
// Uniswapv2Router02 route directs swaps from one pool to another
contract UniswapV2LpTokenCalleeDai is UniswapV2Callee {
}
```

Inherited:

```
// This Callee contract exists as a standalone contract
contract UniswapV2Callee {
    UniswapV2Router02Like   public uniRouter02;
    DaiJoinLike             public daiJoin;
    TokenLike               public dai;

    uint256                 public constant RAY = 10 ** 27;
}
```

13.5.1 add(x, y) [UniswapV2Callee(2)]

```
    function add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

13.5.2 sub(x, y) [UniswapV2Callee(2)]

```
    function sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

13.5.3 divup(x, y) [UniswapV2Callee(2)]

```
    function divup(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = add(x, sub(y, 1)) / y;
    }
```

13.5.4 setUp(uniRouter02_, daiJoin_) [UniswapV2Callee(2)]

```
    function setUp(address uniRouter02_, address daiJoin_) internal {
        uniRouter02 = UniswapV2Router02Like(uniRouter02_);
        daiJoin = DaiJoinLike(daiJoin_);
        dai = daiJoin.dai();

        dai.approve(daiJoin_, uint256(-1));
    }
```

13.5.5 _fromWad(gemJoin, wad) [UniswapV2Callee(2)]

```
    function _fromWad(address gemJoin, uint256 wad) internal view returns (
        ↪ uint256 amt) {
        amt = wad / 10 ** (sub(18, GemJoinLike(gemJoin).dec()));
    }
```

13.5.6 constructor(uniRouter02_, daiJoin_) X

```
    constructor(address uniRouter02_, address daiJoin_) public {
        setUp(uniRouter02_, daiJoin_);
    }
```

13.5.7 swapGemForDai(token, path, to)

```
    function swapGemForDai(
        TokenLike token,
        address[] memory path,
        address to
    ) internal {
        uint256 amountIn = token.balanceOf(address(this));
        token.approve(address(uniRouter02), amountIn);
        uniRouter02.swapExactTokensForTokens(
            amountIn,
            0, // amountOutMin is zero because minProfit is checked at the end
            path,
            address(this),
            block.timestamp
        );
        if (token.balanceOf(address(this)) > 0) {
            token.transfer(to, token.balanceOf(address(this)));
        }
    }
```

13.5.8 clipperCall(sender, daiAmt, gemAmt, data) X

```
    function clipperCall(
        address sender,          // Clipper Caller and Dai deliveryaddress
        uint256 daiAmt,          // Dai amount to payback[rad]
        uint256 gemAmt,          // Gem amount received [wad]
        bytes calldata data      // Extra data needed (gemJoin)
    ) external {
        (
            address to,            // address to send remaining DAI to
            address gemJoin,       // gemJoin adapter address
            uint256 minProfit,     // minimum profit in DAI to make [wad]
            address[] memory pathA, // path of token A
            address[] memory pathB  // path of token B
        ) = abi.decode(data, (address, address, uint256, address[], address[]));

        // Convert gem amount to token precision
        gemAmt = _fromWad(gemJoin, gemAmt);

        // Exit collateral to token version
        GemJoinLike(gemJoin).exit(address(this), gemAmt);

        // Approve uniRouter02 to take gem
        LpTokenLike gem = GemJoinLike(gemJoin).gem();
        gem.approve(address(uniRouter02), gemAmt);

        // Calculate amount of DAI to Join (as erc20 WAD value)
        uint256 daiToJoin = divup(daiAmt, RAY);

        // Do operation and get dai amount bought (checking the profit is
            ↪ achieved)
        TokenLike tokenA = gem.token0();
        TokenLike tokenB = gem.token1();
        uniRouter02.removeLiquidity({ // burn token to obtain its components
            tokenA: address(tokenA),
            tokenB: address(tokenB),
            liquidity: gemAmt,
            amountAMin: 0, // minProfit is checked below
            amountBMin: 0,
            to: address(this),
            deadline: block.timestamp
        });
        if (address(tokenA) != address(dai)) {
            swapGemForDai(tokenA, pathA, to);
        }
        if (address(tokenB) != address(dai)) {
            swapGemForDai(tokenB, pathB, to);
        }
        require(
            dai.balanceOf(address(this)) >= add(daiToJoin, minProfit),
            "UniswapV2Callee/insufficient-profit"
        );

        // Although Uniswap will accept all gems, this check is a sanity check,
            ↪ just in case
        // Transfer any lingering gem to specified address
        if (gem.balanceOf(address(this)) > 0) {
            gem.transfer(to, gem.balanceOf(address(this)));
        }

        // Convert DAI bought to internal vat value of the msg.sender of Clipper
            ↪ .take
        daiJoin.join(sender, daiToJoin);

        // Transfer remaining DAI to specified address
        dai.transfer(to, dai.balanceOf(address(this)));
    }
```

13.6 contract WstETHCurveUniv3Callee

```
contract WstETHCurveUniv3Callee {
    CurvePoolLike   public immutable curvePool;
    UniV3RouterLike public immutable uniV3Router;
    DaiJoinLike     public immutable daiJoin;
    TokenLike       public immutable dai;
    address         public immutable weth;

    uint256         public constant RAY = 10 ** 27;

    receive() external payable {}
}
```

13.6.1 _add(x, y)

```
    function _add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

13.6.2 _sub(x, y)

```
    function _sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

13.6.3 _divup(x, y)

```
    function _divup(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = _add(x, _sub(y, 1)) / y;
    }
```

13.6.4 constructor(curvePool_, uniV3Router_, daiJoin_, weth_) X

```
    constructor(
        address curvePool_,
        address uniV3Router_,
        address daiJoin_,
        address weth_
    ) public {
        curvePool       = CurvePoolLike(curvePool_);
        uniV3Router     = UniV3RouterLike(uniV3Router_);
        daiJoin         = DaiJoinLike(daiJoin_);
        TokenLike dai_  = DaiJoinLike(daiJoin_).dai();
        dai             = dai_;
        weth            = weth_;

        dai_.approve(daiJoin_, type(uint256).max);
    }
```

13.6.5 _fromWad(gemJoin, wad)

```
    function _fromWad(address gemJoin, uint256 wad) internal view returns (
       ↪ uint256 amt) {
        amt = wad / 10 ** (_sub(18, GemJoinLike(gemJoin).dec()));
    }
```

13.6.6 clipperCall(sender, owe, slice, data) X

```
    function clipperCall(
        address sender,             // Clipper caller, pays back the loan
        uint256 owe,                // Dai amount to pay back         [rad]
        uint256 slice,              // Gem amount received            [wad]
        bytes calldata data         // Extra data, see below
    ) external {
        (
            address to,             // address to send remaining DAI to
            address gemJoin,        // gemJoin adapter address
            uint256 minProfit,      // minimum profit in DAI to make [wad]
            bytes memory path,      // uniswap v3 path
            address charterManager // pass address(0) if no manager
        ) = abi.decode(data, (address, address, uint256, bytes, address));

        address gem = GemJoinLike(gemJoin).gem();

        // Convert slice to token precision
        slice = _fromWad(gemJoin, slice);

        // Exit gem to token
        if(charterManager != address(0)) {
            CharterManagerLike(charterManager).exit(gemJoin, address(this),
                ↪ slice);
        } else {
            GemJoinLike(gemJoin).exit(address(this), slice);
        }

        slice = WstEthLike(gem).unwrap(slice);
        gem = WstEthLike(gem).stETH();

        TokenLike(gem).approve(address(curvePool), slice);
        slice = curvePool.exchange({
            i:      1,      // send token id 1 (stETH)
            j:      0,      // receive token id 0 (ETH)
            dx:     slice, // send `slice` amount of stETH
            min_dy: 0       // accept any amount of ETH (`minProfit` is checked
                ↪ below)
        });

        gem = weth;
        WethLike(gem).deposit{
            value: slice
        }();

        // Approve uniV3 to take gem
        WethLike(gem).approve(address(uniV3Router), slice);

        // Calculate amount of DAI to Join (as erc20 WAD value)
        uint256 daiToJoin = _divup(owe, RAY);

        // Do operation and get dai amount bought (checking the profit is
            ↪ achieved)
        UniV3RouterLike.ExactInputParams memory params = UniV3RouterLike.
            ↪ ExactInputParams({
            path:               path,
            recipient:          address(this),
            deadline:           block.timestamp,
            amountIn:           slice,
            amountOutMinimum: _add(daiToJoin, minProfit)
        });
        uniV3Router.exactInput(params);

        // Although Uniswap will accept all gems, this check is a sanity check,
            ↪ just in case
        // Transfer any lingering gem to specified address
        if (WethLike(gem).balanceOf(address(this)) > 0) {
            WethLike(gem).transfer(to, WethLike(gem).balanceOf(address(this)));
        }

        // Convert DAI bought to internal vat value of the msg.sender of Clipper
            ↪ .take
        daiJoin.join(sender, daiToJoin);

        // Transfer remaining DAI to specified address
        dai.transfer(to, dai.balanceOf(address(this)));
    }
```

13.7 contract UniswapV3Callee

```
contract UniswapV3Callee {
    UniV3RouterLike         public uniV3Router;
    DaiJoinLike             public daiJoin;
    TokenLike               public dai;

    uint256                 public constant RAY = 10 ** 27;
}
```

13.7.1 _add(x, y)

```
    function _add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x, "ds-math-add-overflow");
    }
```

13.7.2 _sub(x, y)

```
    function _sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x, "ds-math-sub-underflow");
    }
```

13.7.3 _divup(x, y)

```
    function _divup(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = _add(x, _sub(y, 1)) / y;
    }
```

13.7.4 constructor(uniV3Router_, daiJoin_) X

```
    constructor(address uniV3Router_, address daiJoin_) public {
        uniV3Router = UniV3RouterLike(uniV3Router_);
        daiJoin = DaiJoinLike(daiJoin_);
        dai = daiJoin.dai();

        dai.approve(daiJoin_, uint256(-1));
    }
```

13.7.5 _fromWad(gemJoin, wad)

```
    function _fromWad(address gemJoin, uint256 wad) internal view returns (
       ↪ uint256 amt) {
        amt = wad / 10 ** (_sub(18, GemJoinLike(gemJoin).dec()));
    }
```

13.7.6 clipperCall(sender, owe, slice, data) X

```
    function clipperCall(
        address sender,            // Clipper caller, pays back the loan
        uint256 owe,               // Dai amount to pay back        [rad]
        uint256 slice,             // Gem amount received           [wad]
        bytes calldata data        // Extra data, see below
    ) external {
        (
            address to,            // address to send remaining DAI to
            address gemJoin,       // gemJoin adapter address
            uint256 minProfit,     // minimum profit in DAI to make [wad]
            bytes memory path,     // packed encoding of (address, fee, address
                ↪ [, fee, address?])
            address charterManager // pass address(0) if no manager
        ) = abi.decode(data, (address, address, uint256, bytes, address));

        // Convert slice to token precision
        slice = _fromWad(gemJoin, slice);

        // Exit gem to token
        if(charterManager != address(0)) {
            CharterManagerLike(charterManager).exit(gemJoin, address(this),
                ↪ slice);
        } else {
            GemJoinLike(gemJoin).exit(address(this), slice);
        }

        // Approve uniV3 to take gem
        TokenLike gem = GemJoinLike(gemJoin).gem();
        gem.approve(address(uniV3Router), slice);

        // Calculate amount of DAI to Join (as erc20 WAD value)
        uint256 daiToJoin = _divup(owe, RAY);

        // Do operation and get dai amount bought (checking the profit is
            ↪ achieved)
        UniV3RouterLike.ExactInputParams memory params = UniV3RouterLike.
            ↪ ExactInputParams({
            path:             path,
            recipient:        address(this),
            deadline:         block.timestamp,
            amountIn:         slice,
            amountOutMinimum: _add(daiToJoin, minProfit)
        });
        uniV3Router.exactInput(params);

        // Although Uniswap will accept all gems, this check is a sanity check,
            ↪ just in case
        // Transfer any lingering gem to specified address
        if (gem.balanceOf(address(this)) > 0) {
            gem.transfer(to, gem.balanceOf(address(this)));
        }

        // Convert DAI bought to internal vat value of the msg.sender of Clipper
            ↪ .take
        daiJoin.join(sender, daiToJoin);

        // Transfer remaining DAI to specified address
        dai.transfer(to, dai.balanceOf(address(this)));
    }
```

Chapter 14

Arbitrum bridge

14.1 contract L1DaiGateway

```
contract L1DaiGateway is L1CrossDomainEnabled, L1ITokenGateway {
  // --- Auth ---
  mapping(address => uint256) public wards;

  event Rely(address indexed usr);
  event Deny(address indexed usr);

  address public immutable l1Dai;
  address public immutable l2Dai;
  address public immutable l1Escrow;
  address public immutable l1Router;
  address public immutable l2Counterpart;
  uint256 public isOpen = 1;

  event Closed();
}
```

Inherited:

```
abstract contract L1CrossDomainEnabled {
  IInbox public immutable inbox;

  event TxToL2(address indexed from, address indexed to, uint256 indexed seqNum,
      ↪  bytes data);
}
```

14.1.1 modifier onlyL2Counterpart(l2Counterpart) [L1CrossDomainEnabled]

```
  modifier onlyL2Counterpart(address l2Counterpart) {
    // a message coming from the counterpart gateway was executed by the bridge
    address bridge = inbox.bridge();
    require(msg.sender == bridge, "NOT_FROM_BRIDGE");

    // and the outbox reports that the L2 address of the sender is the
       ↪ counterpart gateway
    address l2ToL1Sender = IOutbox(IBridge(bridge).activeOutbox()).l2ToL1Sender
       ↪ ();
    require(l2ToL1Sender == l2Counterpart, "ONLY_COUNTERPART_GATEWAY");
    _;
  }
```

14.1.2 modifier auth()

```
  modifier auth() {
    require(wards[msg.sender] == 1, "L1DaiGateway/not-authorized");
    _;
  }
```

14.1.3 constructor(_inbox) [L1CrossDomainEnabled] X

```
constructor(address _inbox) public {
  inbox = IInbox(_inbox);
}
```

14.1.4 sendTxToL2(target, user, maxSubmissionCost, maxGas, gasPriceBid, data) [L1CrossDomainEnabled]

```
function sendTxToL2(
  address target,
  address user,
  uint256 maxSubmissionCost,
  uint256 maxGas,
  uint256 gasPriceBid,
  bytes memory data
) internal returns (uint256) {
  uint256 seqNum = inbox.createRetryableTicket{value: msg.value}(
    target,
    0, // we always assume that l2CallValue = 0
    maxSubmissionCost,
    user,
    user,
    maxGas,
    gasPriceBid,
    data
  );
  emit TxToL2(user, target, seqNum, data);
  return seqNum;
}
```

14.1.5 sendTxToL2NoAliasing(target, user, l1CallValue, maxSubmissionCost, maxGas, gasPriceBid, data) [L1CrossDomainEnabled]

```
function sendTxToL2NoAliasing(
  address target,
  address user,
  uint256 l1CallValue,
  uint256 maxSubmissionCost,
  uint256 maxGas,
  uint256 gasPriceBid,
  bytes memory data
) internal returns (uint256) {
  uint256 seqNum = inbox.createRetryableTicketNoRefundAliasRewrite{value:
      ↪ l1CallValue}(
    target,
    0, // we always assume that l2CallValue = 0
    maxSubmissionCost,
    user,
    user,
    maxGas,
    gasPriceBid,
    data
  );
  emit TxToL2(user, target, seqNum, data);
  return seqNum;
}
```

14.1.6 rely(usr) X a

```
function rely(address usr) external auth {
  wards[usr] = 1;
```

```
    emit Rely(usr);
  }
```

14.1.7 deny(usr) X a

```
  function deny(address usr) external auth {
    wards[usr] = 0;
    emit Deny(usr);
  }
```

14.1.8 constructor(_l2Counterpart, _l1Router, _inbox, _l1Dai, _l2Dai, _l1Escrow) X

```
  constructor(
    address _l2Counterpart,
    address _l1Router,
    address _inbox,
    address _l1Dai,
    address _l2Dai,
    address _l1Escrow
  ) public L1CrossDomainEnabled(_inbox) {
    wards[msg.sender] = 1;
    emit Rely(msg.sender);

    l1Dai = _l1Dai;
    l2Dai = _l2Dai;
    l1Escrow = _l1Escrow;
    l1Router = _l1Router;
    l2Counterpart = _l2Counterpart;
  }
```

14.1.9 close() X a

```
  function close() external auth {
    isOpen = 0;

    emit Closed();
  }
```

14.1.10 outboundTransfer(l1Token, to, amount, maxGas, gasPriceBid, data) X

```
  function outboundTransfer(
    address l1Token,
    address to,
    uint256 amount,
    uint256 maxGas,
    uint256 gasPriceBid,
    bytes calldata data
  ) external payable override returns (bytes memory) {
    // do not allow initiating new xchain messages if bridge is closed
    require(isOpen == 1, "L1DaiGateway/closed");
    require(l1Token == l1Dai, "L1DaiGateway/token-not-dai");

    // we use nested scope to avoid stack too deep errors
    address from;
    uint256 seqNum;
    bytes memory extraData;
    {
      uint256 maxSubmissionCost;
      (from, maxSubmissionCost, extraData) = parseOutboundData(data);
      require(extraData.length == 0, "L1DaiGateway/call-hook-data-not-allowed");
      TokenLike(l1Token).transferFrom(from, l1Escrow, amount);

      bytes memory outboundCalldata = getOutboundCalldata(l1Token, from, to,
        ↪ amount, extraData);
      seqNum = sendTxToL2(
        l2Counterpart,
        from,
        maxSubmissionCost,
        maxGas,
        gasPriceBid,
        outboundCalldata
      );
    }

    emit DepositInitiated(l1Token, from, to, seqNum, amount);

    return abi.encode(seqNum);
  }
```

14.1.11 getOutboundCalldata(l1Token, from, to, amount, data)

```
  function getOutboundCalldata(
    address l1Token,
    address from,
    address to,
    uint256 amount,
    bytes memory data
  ) public pure returns (bytes memory outboundCalldata) {
    bytes memory emptyBytes = "";

    outboundCalldata = abi.encodeWithSelector(
      L2ITokenGateway.finalizeInboundTransfer.selector,
      l1Token,
      from,
      to,
      amount,
      abi.encode(emptyBytes, data)
    );

    return outboundCalldata;
  }
```

14.1.12 finalizeInboundTransfer(l1Token, from, to, amount, data) X

```
  function finalizeInboundTransfer(
    address l1Token,
    address from,
    address to,
    uint256 amount,
    bytes calldata data
  ) external override onlyL2Counterpart(l2Counterpart) {
    require(l1Token == l1Dai, "L1DaiGateway/token-not-dai");
    (uint256 exitNum, ) = abi.decode(data, (uint256, bytes));

    TokenLike(l1Token).transferFrom(l1Escrow, to, amount);

    emit WithdrawalFinalized(l1Token, from, to, exitNum, amount);
  }
```

14.1.13 parseOutboundData(data)

```
  function parseOutboundData(bytes memory data)
    internal
    view
    returns (
      address from,
      uint256 maxSubmissionCost,
      bytes memory extraData
    )
  {
    if (msg.sender == l1Router) {
      // router encoded
      (from, extraData) = abi.decode(data, (address, bytes));
    } else {
      from = msg.sender;
      extraData = data;
    }
    // user encoded
    (maxSubmissionCost, extraData) = abi.decode(extraData, (uint256, bytes));
  }
```

14.1.14 calculateL2TokenAddress(l1Token)

```
  function calculateL2TokenAddress(address l1Token) external view override
      ↪ returns (address) {
    if (l1Token != l1Dai) {
      return address(0);
    }

    return l2Dai;
  }
```

14.1.15 counterpartGateway()

```
  function counterpartGateway() external view override returns (address) {
    return l2Counterpart;
  }
```

14.2 contract L1Escrow

```
// Escrow funds on L1, manage approval rights

contract L1Escrow {
  // --- Auth ---
  mapping(address => uint256) public wards;

  event Rely(address indexed usr);
  event Deny(address indexed usr);

  event Approve(address indexed token, address indexed spender, uint256 value);
}
```

14.2.1 modifier auth()

```
  modifier auth() {
    require(wards[msg.sender] == 1, "L1Escrow/not-authorized");
    _;
  }
```

14.2.2 rely(usr) X a

```
  function rely(address usr) external auth {
    wards[usr] = 1;
    emit Rely(usr);
  }
```

14.2.3 deny(usr) X a

```
  function deny(address usr) external auth {
    wards[usr] = 0;
    emit Deny(usr);
  }
```

14.2.4 constructor() X

```
  constructor() public {
    wards[msg.sender] = 1;
    emit Rely(msg.sender);
  }
```

14.2.5 approve(token, spender, value) X a

```
  function approve(
    address token,
    address spender,
    uint256 value
  ) external auth {
    emit Approve(token, spender, value);

    ApproveLike(token).approve(spender, value);
  }
```

14.3 contract L1GovernanceRelay

```
// Relay a message from L1 to L2GovernanceRelay
// Sending L1->L2 message on arbitrum requires ETH balance. That's why this
    ↪ contract can receive ether.
// Excessive ether can be reclaimed by governance by calling reclaim function.

contract L1GovernanceRelay is L1CrossDomainEnabled {
  // --- Auth ---
  mapping(address => uint256) public wards;

  address public immutable l2GovernanceRelay;

  event Rely(address indexed usr);
  event Deny(address indexed usr);

  // Allow contract to receive ether
  receive() external payable {}
}
```

Inherited:

```
abstract contract L1CrossDomainEnabled {
  IInbox public immutable inbox;

  event TxToL2(address indexed from, address indexed to, uint256 indexed seqNum,
      ↪  bytes data);
}
```

14.3.1 modifier onlyL2Counterpart(l2Counterpart) [L1CrossDomainEnabled]

```
  modifier onlyL2Counterpart(address l2Counterpart) {
    // a message coming from the counterpart gateway was executed by the bridge
    address bridge = inbox.bridge();
    require(msg.sender == bridge, "NOT_FROM_BRIDGE");

    // and the outbox reports that the L2 address of the sender is the
       ↪ counterpart gateway
    address l2ToL1Sender = IOutbox(IBridge(bridge).activeOutbox()).l2ToL1Sender
       ↪ ();
    require(l2ToL1Sender == l2Counterpart, "ONLY_COUNTERPART_GATEWAY");
    _;
  }
```

14.3.2 modifier auth()

```
  modifier auth() {
    require(wards[msg.sender] == 1, "L1GovernanceRelay/not-authorized");
    _;
  }
```

14.3.3 constructor(_inbox) [L1CrossDomainEnabled] X

```
  constructor(address _inbox) public {
    inbox = IInbox(_inbox);
  }
```

14.3.4 sendTxToL2(target, user, maxSubmissionCost, maxGas, gasPriceBid, data) [L1CrossDomainEnabled]

```
function sendTxToL2(
  address target,
  address user,
  uint256 maxSubmissionCost,
  uint256 maxGas,
  uint256 gasPriceBid,
  bytes memory data
) internal returns (uint256) {
  uint256 seqNum = inbox.createRetryableTicket{value: msg.value}(
    target,
    0, // we always assume that l2CallValue = 0
    maxSubmissionCost,
    user,
    user,
    maxGas,
    gasPriceBid,
    data
  );
  emit TxToL2(user, target, seqNum, data);
  return seqNum;
}
```

14.3.5 sendTxToL2NoAliasing(target, user, l1CallValue, maxSubmissionCost, maxGas, gasPriceBid, data) [L1CrossDomainEnabled]

```
function sendTxToL2NoAliasing(
  address target,
  address user,
  uint256 l1CallValue,
  uint256 maxSubmissionCost,
  uint256 maxGas,
  uint256 gasPriceBid,
  bytes memory data
) internal returns (uint256) {
  uint256 seqNum = inbox.createRetryableTicketNoRefundAliasRewrite{value:
      ↪ l1CallValue}(
    target,
    0, // we always assume that l2CallValue = 0
    maxSubmissionCost,
    user,
    user,
    maxGas,
    gasPriceBid,
    data
  );
  emit TxToL2(user, target, seqNum, data);
  return seqNum;
}
```

14.3.6 rely(usr) X a

```
function rely(address usr) external auth {
  wards[usr] = 1;
  emit Rely(usr);
}
```

14.3.7 deny(usr) X a

```
function deny(address usr) external auth {
  wards[usr] = 0;
  emit Deny(usr);
}
```

14.3.8 constructor(_inbox, _l2GovernanceRelay) X

```
constructor(address _inbox, address _l2GovernanceRelay) public
    ↪ L1CrossDomainEnabled(_inbox) {
  wards[msg.sender] = 1;
  emit Rely(msg.sender);

  l2GovernanceRelay = _l2GovernanceRelay;
}
```

14.3.9 reclaim(receiver, amount) X a

```
// Allow governance to reclaim stored ether
function reclaim(address receiver, uint256 amount) external auth {
  (bool sent, ) = receiver.call{value: amount}("");
  require(sent, "L1GovernanceRelay/failed-to-send-ether");
}
```

14.3.10 relay(target, targetData, l1CallValue, maxGas, gasPriceBid, maxSubmissionCost) X a

```
// Forward a call to be repeated on L2
function relay(
  address target,
  bytes calldata targetData,
  uint256 l1CallValue,
  uint256 maxGas,
  uint256 gasPriceBid,
  uint256 maxSubmissionCost
) external payable auth {
  bytes memory data = abi.encodeWithSelector(
    L2GovernanceRelay.relay.selector,
    target,
    targetData
  );

  sendTxToL2NoAliasing(
    l2GovernanceRelay,
    l2GovernanceRelay, // send any excess ether to the L2 counterpart
    l1CallValue,
    maxSubmissionCost,
    maxGas,
    gasPriceBid,
    data
  );
}
```

14.4 contract L2DaiGateway

```
contract L2DaiGateway is L2CrossDomainEnabled, L2ITokenGateway {
  // --- Auth ---
  mapping(address => uint256) public wards;

  event Rely(address indexed usr);
  event Deny(address indexed usr);

  address public immutable l1Dai;
  address public immutable l2Dai;
  address public immutable l1Counterpart;
  address public immutable l2Router;
  uint256 public isOpen = 1;

  event Closed();
}
```

Inherited:

```
abstract contract L2CrossDomainEnabled {
  event TxToL1(address indexed from, address indexed to, uint256 indexed id,
      ↪ bytes data);

  uint160 constant offset = uint160(0x1111000000000000000000000000000000001111);
}
```

14.4.1 modifier onlyL1Counterpart(l1Counterpart) [L2CrossDomainEnabled]

```
  modifier onlyL1Counterpart(address l1Counterpart) {
    require(msg.sender == applyL1ToL2Alias(l1Counterpart), "
      ↪ ONLY_COUNTERPART_GATEWAY");
    _;
  }
```

14.4.2 modifier auth()

```
  modifier auth() {
    require(wards[msg.sender] == 1, "L2DaiGateway/not-authorized");
    _;
  }
```

14.4.3 sendTxToL1(user, to, data) [L2CrossDomainEnabled]

```
  function sendTxToL1(
    address user,
    address to,
    bytes memory data
  ) internal returns (uint256) {
    // note: this method doesn't support sending ether to L1 together with a
      ↪ call
    uint256 id = ArbSys(address(100)).sendTxToL1(to, data);

    emit TxToL1(user, to, id, data);

    return id;
  }
```

14.4.4 applyL1ToL2Alias(l1Address) [L2CrossDomainEnabled]

```
// l1 addresses are transformed durng l1->l2 calls
function applyL1ToL2Alias(address l1Address) internal pure returns (address
    ↪ l2Address) {
  l2Address = address(uint160(l1Address) + offset);
}
```

14.4.5 rely(usr) X a

```
function rely(address usr) external auth {
  wards[usr] = 1;
  emit Rely(usr);
}
```

14.4.6 deny(usr) X a

```
function deny(address usr) external auth {
  wards[usr] = 0;
  emit Deny(usr);
}
```

14.4.7 constructor(_l1Counterpart, _l2Router, _l1Dai, _l2Dai) X

```
constructor(
  address _l1Counterpart,
  address _l2Router,
  address _l1Dai,
  address _l2Dai
) public {
  wards[msg.sender] = 1;
  emit Rely(msg.sender);

  l1Dai = _l1Dai;
  l2Dai = _l2Dai;
  l1Counterpart = _l1Counterpart;
  l2Router = _l2Router;
}
```

14.4.8 close() X a

```
function close() external auth {
  isOpen = 0;

  emit Closed();
}
```

14.4.9 outboundTransfer(l1Token, to, amount, data) X

```
function outboundTransfer(
  address l1Token,
  address to,
  uint256 amount,
  bytes calldata data
) external returns (bytes memory) {
  return outboundTransfer(l1Token, to, amount, 0, 0, data);
}
```

14.4.10 outboundTransfer(l1Token, to, amount, , , data) X

```
function outboundTransfer(
  address l1Token,
  address to,
  uint256 amount,
  uint256, // maxGas
  uint256, // gasPriceBid
  bytes calldata data
) public override returns (bytes memory res) {
  require(isOpen == 1, "L2DaiGateway/closed");
  require(l1Token == l1Dai, "L2DaiGateway/token-not-dai");

  (address from, bytes memory extraData) = parseOutboundData(data);
  require(extraData.length == 0, "L2DaiGateway/call-hook-data-not-allowed");

  Mintable(l2Dai).burn(from, amount);

  uint256 id = sendTxToL1(
    from,
    l1Counterpart,
    getOutboundCalldata(l1Token, from, to, amount, extraData)
  );

  // we don't need to track exitNums (b/c we have no fast exits) so we always
     ↪ use 0
  emit WithdrawalInitiated(l1Token, from, to, id, 0, amount);

  return abi.encode(id);
}
```

14.4.11 getOutboundCalldata(token, from, to, amount, data)

```
function getOutboundCalldata(
  address token,
  address from,
  address to,
  uint256 amount,
  bytes memory data
) public pure returns (bytes memory outboundCalldata) {
  outboundCalldata = abi.encodeWithSelector(
    L1ITokenGateway.finalizeInboundTransfer.selector,
    token,
    from,
    to,
    amount,
    abi.encode(0, data) // we don't need to track exitNums (b/c we have no
       ↪ fast exits) so we always use 0
  );

  return outboundCalldata;
}
```

14.4.12 finalizeInboundTransfer(l1Token, from, to, amount,) X

```
function finalizeInboundTransfer(
  address l1Token,
  address from,
  address to,
  uint256 amount,
  bytes calldata // data -- unsused
) external override onlyL1Counterpart(l1Counterpart) {
  require(l1Token == l1Dai, "L2DaiGateway/token-not-dai");

  Mintable(l2Dai).mint(to, amount);
```

```
    emit DepositFinalized(l1Token, from, to, amount);
  }
```

14.4.13 calculateL2TokenAddress(l1Token)

```
  function calculateL2TokenAddress(address l1Token) external view override
      ↪ returns (address) {
    if (l1Token != l1Dai) {
      return address(0);
    }

    return l2Dai;
  }
```

14.4.14 parseOutboundData(data)

```
  function parseOutboundData(bytes memory data)
    internal
    view
    returns (address from, bytes memory extraData)
  {
    if (msg.sender == l2Router) {
      (from, extraData) = abi.decode(data, (address, bytes));
    } else {
      from = msg.sender;
      extraData = data;
    }
  }
```

14.4.15 counterpartGateway()

```
  function counterpartGateway() external view override returns (address) {
    return l1Counterpart;
  }
```

14.5 contract L2GovernanceRelay

```
// Receive xchain message from L1 counterpart and execute given spell

contract L2GovernanceRelay is L2CrossDomainEnabled {
  address public immutable l1GovernanceRelay;

  // Allow contract to receive ether
  receive() external payable {}
}
```

Inherited:

```
abstract contract L2CrossDomainEnabled {
  event TxToL1(address indexed from, address indexed to, uint256 indexed id,
     ↪ bytes data);

  uint160 constant offset = uint160(0x1111000000000000000000000000000000001111);
}
```

14.5.1 modifier onlyL1Counterpart(l1Counterpart) [L2CrossDomainEnabled]

```
  modifier onlyL1Counterpart(address l1Counterpart) {
    require(msg.sender == applyL1ToL2Alias(l1Counterpart), "
       ↪ ONLY_COUNTERPART_GATEWAY");
    _;
  }
```

14.5.2 sendTxToL1(user, to, data) [L2CrossDomainEnabled]

```
  function sendTxToL1(
    address user,
    address to,
    bytes memory data
  ) internal returns (uint256) {
    // note: this method doesn't support sending ether to L1 together with a
       ↪ call
    uint256 id = ArbSys(address(100)).sendTxToL1(to, data);

    emit TxToL1(user, to, id, data);

    return id;
  }
```

14.5.3 applyL1ToL2Alias(l1Address) [L2CrossDomainEnabled]

```
  // l1 addresses are transformed durng l1->l2 calls
  function applyL1ToL2Alias(address l1Address) internal pure returns (address
     ↪ l2Address) {
    l2Address = address(uint160(l1Address) + offset);
  }
```

14.5.4 constructor(_l1GovernanceRelay) X

```
  constructor(address _l1GovernanceRelay) public {
    l1GovernanceRelay = _l1GovernanceRelay;
  }
```

14.5.5 relay(target, targetData) X

```
function relay(address target, bytes calldata targetData)
  external
  onlyL1Counterpart(l1GovernanceRelay)
{
  (bool ok, ) = target.delegatecall(targetData);
  // note: even if a retryable call fails, it can be retried
  require(ok, "L2GovernanceRelay/delegatecall-error");
}
```

14.6 contract Dai(2)

```
// Improved Dai token

contract Dai {

  // --- Auth ---
  mapping (address => uint256) public wards;

  // --- ERC20 Data ---
  string  public constant name     = "Dai Stablecoin";
  string  public constant symbol   = "DAI";
  string  public constant version  = "2";
  uint8   public constant decimals = 18;
  uint256 public totalSupply;

  mapping (address => uint256)                      public balanceOf;
  mapping (address => mapping (address => uint256)) public allowance;
  mapping (address => uint256)                      public nonces;

  event Approval(address indexed owner, address indexed spender, uint256 value);
  event Transfer(address indexed from, address indexed to, uint256 value);
  event Rely(address indexed usr);
  event Deny(address indexed usr);

  // --- EIP712 niceties ---
  uint256 public immutable deploymentChainId;
  bytes32 private immutable _DOMAIN_SEPARATOR;
  bytes32 public constant PERMIT_TYPEHASH = keccak256("Permit(address owner,
     ↪ address spender,uint256 value,uint256 nonce,uint256 deadline)");
}
```

14.6.1 modifier auth()

```
  modifier auth {
    require(wards[msg.sender] == 1, "Dai/not-authorized");
    _;
  }
```

14.6.2 rely(usr) X a

```
  function rely(address usr) external auth {
    wards[usr] = 1;
    emit Rely(usr);
  }
```

14.6.3 deny(usr) X a

```
  function deny(address usr) external auth {
    wards[usr] = 0;
    emit Deny(usr);
  }
```

14.6.4 _add(x, y)

```
  // --- Math ---
  function _add(uint256 x, uint256 y) internal pure returns (uint256 z) {
    require((z = x + y) >= x);
  }
```

14.6.5 _sub(x, y)

```
function _sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
  require((z = x - y) <= x);
}
```

14.6.6 constructor() X

```
constructor() public {
  wards[msg.sender] = 1;
  emit Rely(msg.sender);

  uint256 chainId;
  assembly {chainId := chainid()}
  deploymentChainId = chainId;
  _DOMAIN_SEPARATOR = _calculateDomainSeparator(chainId);
}
```

14.6.7 _calculateDomainSeparator(chainId)

```
function _calculateDomainSeparator(uint256 chainId) private view returns (
    ↪ bytes32) {
  return keccak256(
    abi.encode(
      keccak256("EIP712Domain(string name,string version,uint256 chainId,
          ↪ address verifyingContract)"),
      keccak256(bytes(name)),
      keccak256(bytes(version)),
      chainId,
      address(this)
    )
  );
}
```

14.6.8 DOMAIN_SEPARATOR()

```
function DOMAIN_SEPARATOR() external view returns (bytes32) {
  uint256 chainId;
  assembly {chainId := chainid()}
  return chainId == deploymentChainId ? _DOMAIN_SEPARATOR :
      ↪ _calculateDomainSeparator(chainId);
}
```

14.6.9 transfer(to, value) X

```
// --- ERC20 Mutations ---
function transfer(address to, uint256 value) external returns (bool) {
  require(to != address(0) && to != address(this), "Dai/invalid-address");
  uint256 balance = balanceOf[msg.sender];
  require(balance >= value, "Dai/insufficient-balance");

  balanceOf[msg.sender] = balance - value;
  balanceOf[to] += value;

  emit Transfer(msg.sender, to, value);

  return true;
}
```

14.6.10 transferFrom(from, to, value) X

```
function transferFrom(address from, address to, uint256 value) external
    ↪ returns (bool) {
  require(to != address(0) && to != address(this), "Dai/invalid-address");
  uint256 balance = balanceOf[from];
  require(balance >= value, "Dai/insufficient-balance");

  if (from != msg.sender) {
    uint256 allowed = allowance[from][msg.sender];
    if (allowed != type(uint256).max) {
      require(allowed >= value, "Dai/insufficient-allowance");

      allowance[from][msg.sender] = allowed - value;
    }
  }

  balanceOf[from] = balance - value;
  balanceOf[to] += value;

  emit Transfer(from, to, value);

  return true;
}
```

14.6.11 approve(spender, value) X

```
function approve(address spender, uint256 value) external returns (bool) {
  allowance[msg.sender][spender] = value;

  emit Approval(msg.sender, spender, value);

  return true;
}
```

14.6.12 increaseAllowance(spender, addedValue) X

```
function increaseAllowance(address spender, uint256 addedValue) external
    ↪ returns (bool) {
  uint256 newValue = _add(allowance[msg.sender][spender], addedValue);
  allowance[msg.sender][spender] = newValue;

  emit Approval(msg.sender, spender, newValue);

  return true;
}
```

14.6.13 decreaseAllowance(spender, subtractedValue) X

```
function decreaseAllowance(address spender, uint256 subtractedValue) external
    ↪ returns (bool) {
  uint256 allowed = allowance[msg.sender][spender];
  require(allowed >= subtractedValue, "Dai/insufficient-allowance");
  allowed = allowed - subtractedValue;
  allowance[msg.sender][spender] = allowed;

  emit Approval(msg.sender, spender, allowed);

  return true;
}
```

14.6.14 mint(to, value) X a

```
// --- Mint/Burn ---
function mint(address to, uint256 value) external auth {
  require(to != address(0) && to != address(this), "Dai/invalid-address");
  balanceOf[to] = balanceOf[to] + value; // note: we don't need an overflow
     ↪ check here b/c balanceOf[to] <= totalSupply and there is an overflow
     ↪ check below
  totalSupply   = _add(totalSupply, value);

  emit Transfer(address(0), to, value);
}
```

14.6.15 burn(from, value) X

```
function burn(address from, uint256 value) external {
  uint256 balance = balanceOf[from];
  require(balance >= value, "Dai/insufficient-balance");

  if (from != msg.sender && wards[msg.sender] != 1) {
    uint256 allowed = allowance[from][msg.sender];
    if (allowed != type(uint256).max) {
      require(allowed >= value, "Dai/insufficient-allowance");

      allowance[from][msg.sender] = allowed - value;
    }
  }

  balanceOf[from] = balance - value; // note: we don't need overflow checks b/
     ↪ c require(balance >= value) and balance <= totalSupply
  totalSupply     = totalSupply - value;

  emit Transfer(from, address(0), value);
}
```

14.6.16 permit(owner, spender, value, deadline, v, r, s) X

```
// --- Approve by signature ---
function permit(address owner, address spender, uint256 value, uint256
    ↪ deadline, uint8 v, bytes32 r, bytes32 s) external {
  require(block.timestamp <= deadline, "Dai/permit-expired");

  uint256 chainId;
  assembly {chainId := chainid()}

  bytes32 digest =
    keccak256(abi.encodePacked(
        "\x19\x01",
        chainId == deploymentChainId ? _DOMAIN_SEPARATOR :
            ↪ _calculateDomainSeparator(chainId),
        keccak256(abi.encode(
          PERMIT_TYPEHASH,
          owner,
          spender,
          value,
          nonces[owner]++,
          deadline
        ))
    ));

  require(owner != address(0) && owner == ecrecover(digest, v, r, s), "Dai/
     ↪ invalid-permit");

  allowance[owner][spender] = value;
  emit Approval(owner, spender, value);
}
```

Chapter 15

Optimism bridge

15.1 contract L1DAITokenBridge

```
// Managed locked funds in L1Escrow and send / receive messages to
    ↪ L2DAITokenBridge counterpart
// Note: when bridge is closed it will still process in progress messages

contract L1DAITokenBridge is iOVM_L1ERC20Bridge, OVM_CrossDomainEnabled {
  // --- Auth ---
  mapping(address => uint256) public wards;

  event Rely(address indexed usr);
  event Deny(address indexed usr);

  address public immutable l1Token;
  address public immutable l2DAITokenBridge;
  address public immutable l2Token;
  address public immutable escrow;
  uint256 public isOpen = 1;

  event Closed();
}
```

Inherited:

```
/**
 * @title OVM_CrossDomainEnabled
 * @dev Helper contract for contracts performing cross-domain communications
 *
 * Compiler used: defined by inheriting contract
 * Runtime target: defined by inheriting contract
 */
contract OVM_CrossDomainEnabled {

    /*************
     * Variables *
     *************/

    // Messenger contract used to send and recieve messages from the other
        ↪ domain.
    address public messenger;


    /***************
     * Constructor *
     ***************/


    /**********************
     * Function Modifiers *
     **********************/


    /**********************
     * Internal Functions *
     **********************/
}
```

15.1.1 modifier onlyFromCrossDomainAccount(_sourceDomainAccount) [OVM_CrossDomainEnabled]

```
    /**
     * Enforces that the modified function is only callable by a specific cross-
        ↪ domain account.
     * @param _sourceDomainAccount The only account on the originating domain
        ↪ which is
     *  authenticated to call this function.
     */
    modifier onlyFromCrossDomainAccount(
        address _sourceDomainAccount
    ) {
        require(
            msg.sender == address(getCrossDomainMessenger()),
            "OVM_XCHAIN: messenger contract unauthenticated"
        );

        require(
            getCrossDomainMessenger().xDomainMessageSender() ==
                ↪ _sourceDomainAccount,
            "OVM_XCHAIN: wrong sender of cross-domain message"
        );

        _;
    }
```

15.1.2 modifier auth()

```
  modifier auth() {
    require(wards[msg.sender] == 1, "L1DAITokenBridge/not-authorized");
    _;
  }
```

15.1.3 constructor(_messenger) [OVM_CrossDomainEnabled] X

```
    /**
     * @param _messenger Address of the CrossDomainMessenger on the current
        ↪ layer.
     */
    constructor(
        address _messenger
    ) {
        messenger = _messenger;
    }
```

15.1.4 getCrossDomainMessenger() [OVM_CrossDomainEnabled]

```
    /**
     * Gets the messenger, usually from storage. This function is exposed in
        ↪ case a child contract
     * needs to override.
     * @return The address of the cross-domain messenger contract which should
        ↪ be used.
     */
    function getCrossDomainMessenger()
        internal
        virtual
        returns (
            iOVM_CrossDomainMessenger
        )
    {
        return iOVM_CrossDomainMessenger(messenger);
    }
```

15.1.5 sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message) [OVM_CrossDomainEnabled]

```
    /**
     * Sends a message to an account on another domain
     * @param _crossDomainTarget The intended recipient on the destination
        ↪ domain
     * @param _message The data to send to the target (usually calldata to a
        ↪ function with
     *  ‘onlyFromCrossDomainAccount()‘)
     * @param _gasLimit The gasLimit for the receipt of the message on the
        ↪ target domain.
     */
    function sendCrossDomainMessage(
        address _crossDomainTarget,
        uint32 _gasLimit,
        bytes memory _message
    )
        internal
    {
        getCrossDomainMessenger().sendMessage(_crossDomainTarget, _message,
           ↪ _gasLimit);
    }
```

15.1.6 rely(usr) X a

```
  function rely(address usr) external auth {
    wards[usr] = 1;
    emit Rely(usr);
  }
```

15.1.7 deny(usr) X a

```
  function deny(address usr) external auth {
    wards[usr] = 0;
    emit Deny(usr);
  }
```

15.1.8 constructor(_l1Token, _l2DAITokenBridge, _l2Token, _l1messenger, _escrow) X

```
  constructor(
    address _l1Token,
    address _l2DAITokenBridge,
    address _l2Token,
    address _l1messenger,
    address _escrow
  ) OVM_CrossDomainEnabled(_l1messenger) {
    wards[msg.sender] = 1;
    emit Rely(msg.sender);

    l1Token = _l1Token;
    l2DAITokenBridge = _l2DAITokenBridge;
    l2Token = _l2Token;
    escrow = _escrow;
  }
```

15.1.9 close() X a

```
  function close() external auth {
    isOpen = 0;

    emit Closed();
  }
```

15.1.10 depositERC20(_l1Token, _l2Token, _amount, _l2Gas, _data) X

```
  function depositERC20(
    address _l1Token,
    address _l2Token,
    uint256 _amount,
    uint32 _l2Gas,
    bytes calldata _data
  ) external virtual override {
    // Used to stop deposits from contracts (avoid accidentally lost tokens)
    // Note: This check could be bypassed by a malicious contract via initcode,
        ↪ but it takes care of the user error we want to avoid.
    require(!Address.isContract(msg.sender), "L1DAITokenBridge/Sender-not-EOA");
    require(_l1Token == l1Token && _l2Token == l2Token, "L1DAITokenBridge/token-
        ↪ not-dai");

    _initiateERC20Deposit(msg.sender, msg.sender, _amount, _l2Gas, _data);
  }
```

15.1.11 depositERC20To(_l1Token, _l2Token, _to, _amount, _l2Gas, _data) X

```
  function depositERC20To(
    address _l1Token,
    address _l2Token,
    address _to,
    uint256 _amount,
    uint32 _l2Gas,
    bytes calldata _data
  ) external virtual override {
    require(_l1Token == l1Token && _l2Token == l2Token, "L1DAITokenBridge/token-
        ↪ not-dai");

    _initiateERC20Deposit(msg.sender, _to, _amount, _l2Gas, _data);
  }
```

15.1.12 _initiateERC20Deposit(_from, _to, _amount, _l2Gas, _data)

```
  function _initiateERC20Deposit(
    address _from,
    address _to,
    uint256 _amount,
    uint32 _l2Gas,
    bytes calldata _data
  ) internal {
    // do not allow initiating new xchain messages if bridge is closed
    require(isOpen == 1, "L1DAITokenBridge/closed");

    TokenLike(l1Token).transferFrom(_from, escrow, _amount);

    bytes memory message = abi.encodeWithSelector(
      iOVM_L2ERC20Bridge.finalizeDeposit.selector,
      l1Token,
      l2Token,
      _from,
      _to,
      _amount,
      _data
    );

    sendCrossDomainMessage(l2DAITokenBridge, _l2Gas, message);

    emit ERC20DepositInitiated(l1Token, l2Token, _from, _to, _amount, _data);
  }
```

15.1.13 finalizeERC20Withdrawal(_l1Token, _l2Token, _from, _to, _amount, _data) X

```
  function finalizeERC20Withdrawal(
    address _l1Token,
    address _l2Token,
    address _from,
    address _to,
    uint256 _amount,
    bytes calldata _data
  ) external override onlyFromCrossDomainAccount(l2DAITokenBridge) {
    require(_l1Token == l1Token && _l2Token == l2Token, "L1DAITokenBridge/token-
       ↪ not-dai");

    TokenLike(l1Token).transferFrom(escrow, _to, _amount);

    emit ERC20WithdrawalFinalized(l1Token, l2Token, _from, _to, _amount, _data);
  }
```

15.2 contract L1Escrow(2)

```
// Escrow funds on L1, manage approval rights

contract L1Escrow {
  // --- Auth ---
  mapping(address => uint256) public wards;

  event Rely(address indexed usr);
  event Deny(address indexed usr);

  event Approve(address indexed token, address indexed spender, uint256 value);
}
```

15.2.1 modifier auth()

```
  modifier auth() {
    require(wards[msg.sender] == 1, "L1Escrow/not-authorized");
    _;
  }
```

15.2.2 rely(usr) X a

```
  function rely(address usr) external auth {
    wards[usr] = 1;
    emit Rely(usr);
  }
```

15.2.3 deny(usr) X a

```
  function deny(address usr) external auth {
    wards[usr] = 0;
    emit Deny(usr);
  }
```

15.2.4 constructor() X

```
  constructor() {
    wards[msg.sender] = 1;
    emit Rely(msg.sender);
  }
```

15.2.5 approve(token, spender, value) X a

```
  function approve(
    address token,
    address spender,
    uint256 value
  ) external auth {
    emit Approve(token, spender, value);

    ApproveLike(token).approve(spender, value);
  }
```

15.3 contract L1GovernanceRelay(2)

```
// Relay a message from L1 to L2GovernanceRelay

contract L1GovernanceRelay is OVM_CrossDomainEnabled {
  // --- Auth ---
  mapping(address => uint256) public wards;

  address public immutable l2GovernanceRelay;

  event Rely(address indexed usr);
  event Deny(address indexed usr);
}
```

Inherited:

```
/**
 * @title OVM_CrossDomainEnabled
 * @dev Helper contract for contracts performing cross-domain communications
 *
 * Compiler used: defined by inheriting contract
 * Runtime target: defined by inheriting contract
 */
contract OVM_CrossDomainEnabled {

    /*************
     * Variables *
     *************/

    // Messenger contract used to send and recieve messages from the other
        ↪ domain.
    address public messenger;


    /***************
     * Constructor *
     ***************/


    /**********************
     * Function Modifiers *
     **********************/


    /**********************
     * Internal Functions *
     **********************/
}
```

15.3.1 modifier onlyFromCrossDomainAccount(_sourceDomainAccount) [OVM_CrossDomainEnabled]

```
    /**
     * Enforces that the modified function is only callable by a specific cross-
         ↪ domain account.
     * @param _sourceDomainAccount The only account on the originating domain
         ↪ which is
     *  authenticated to call this function.
     */
    modifier onlyFromCrossDomainAccount(
        address _sourceDomainAccount
    ) {
        require(
            msg.sender == address(getCrossDomainMessenger()),
            "OVM_XCHAIN: messenger contract unauthenticated"
        );
        require(
            getCrossDomainMessenger().xDomainMessageSender() ==
                ↪ _sourceDomainAccount,
            "OVM_XCHAIN: wrong sender of cross-domain message"
        );

        _;
    }
```

15.3.2 modifier auth()

```
  modifier auth() {
    require(wards[msg.sender] == 1, "L1GovernanceRelay/not-authorized");
    _;
  }
```

15.3.3 constructor(_messenger) [OVM_CrossDomainEnabled] X

```
    /**
     * @param _messenger Address of the CrossDomainMessenger on the current
         ↪ layer.
     */
    constructor(
        address _messenger
    ) {
        messenger = _messenger;
    }
```

15.3.4 getCrossDomainMessenger() [OVM_CrossDomainEnabled]

```
    /**
     * Gets the messenger, usually from storage. This function is exposed in
         ↪ case a child contract
     * needs to override.
     * @return The address of the cross-domain messenger contract which should
         ↪ be used.
     */
    function getCrossDomainMessenger()
        internal
        virtual
        returns (
            iOVM_CrossDomainMessenger
        )
    {
        return iOVM_CrossDomainMessenger(messenger);
    }
```

15.3.5 sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message) [OVM_CrossDomainEnabled]

```
    /**
     * Sends a message to an account on another domain
     * @param _crossDomainTarget The intended recipient on the destination
         ↪ domain
     * @param _message The data to send to the target (usually calldata to a
         ↪ function with
     *  `onlyFromCrossDomainAccount()`)
     * @param _gasLimit The gasLimit for the receipt of the message on the
         ↪ target domain.
     */
    function sendCrossDomainMessage(
        address _crossDomainTarget,
        uint32 _gasLimit,
        bytes memory _message
    )
        internal
    {
        getCrossDomainMessenger().sendMessage(_crossDomainTarget, _message,
            ↪ _gasLimit);
    }
```

15.3.6 rely(usr) X a

```
function rely(address usr) external auth {
    wards[usr] = 1;
    emit Rely(usr);
}
```

15.3.7 deny(usr) X a

```
function deny(address usr) external auth {
    wards[usr] = 0;
    emit Deny(usr);
}
```

15.3.8 constructor(_l2GovernanceRelay, _l1messenger) X

```
constructor(address _l2GovernanceRelay, address _l1messenger)
    OVM_CrossDomainEnabled(_l1messenger)
{
    wards[msg.sender] = 1;
    emit Rely(msg.sender);

    l2GovernanceRelay = _l2GovernanceRelay;
}
```

15.3.9 relay(target, targetData, l2gas) X a

```
// Forward a call to be repeated on L2
function relay(
    address target,
    bytes calldata targetData,
    uint32 l2gas
) external auth {
    bytes memory data = abi.encodeWithSelector(
        L2GovernanceRelay.relay.selector,
        target,
        targetData
    );

    sendCrossDomainMessage(l2GovernanceRelay, l2gas, data);
}
```

15.4 contract L2DAITokenBridge

```
// Mint tokens on L2 after locking funds on L1.
// Burn tokens on L1 and send a message to unlock tokens on L1 to L1 counterpart
// Note: when bridge is closed it will still process in progress messages

contract L2DAITokenBridge is iOVM_L2ERC20Bridge, OVM_CrossDomainEnabled {
  // --- Auth ---
  mapping(address => uint256) public wards;

  event Rely(address indexed usr);
  event Deny(address indexed usr);

  address public immutable l1Token;
  address public immutable l2Token;
  address public immutable l1DAITokenBridge;
  uint256 public isOpen = 1;

  event Closed();
}
```

Inherited:

```
/**
 * @title OVM_CrossDomainEnabled
 * @dev Helper contract for contracts performing cross-domain communications
 *
 * Compiler used: defined by inheriting contract
 * Runtime target: defined by inheriting contract
 */
contract OVM_CrossDomainEnabled {

    /*************
     * Variables *
     *************/

    // Messenger contract used to send and recieve messages from the other
       ↪ domain.
    address public messenger;


    /***************
     * Constructor *
     ***************/


    /********************
     * Function Modifiers *
     ********************/


    /********************
     * Internal Functions *
     ********************/
}
```

15.4.1 modifier onlyFromCrossDomainAccount(_sourceDomainAccount) [OVM_CrossDomainEnabled]

```
    /**
     * Enforces that the modified function is only callable by a specific cross-
       ↪ domain account.
     * @param _sourceDomainAccount The only account on the originating domain
       ↪ which is
     *  authenticated to call this function.
     */
    modifier onlyFromCrossDomainAccount(
        address _sourceDomainAccount
    ) {
        require(
            msg.sender == address(getCrossDomainMessenger()),
            "OVM_XCHAIN: messenger contract unauthenticated"
        );

        require(
            getCrossDomainMessenger().xDomainMessageSender() ==
                ↪ _sourceDomainAccount,
            "OVM_XCHAIN: wrong sender of cross-domain message"
        );

        _;
    }
```

15.4.2 modifier auth()

```
  modifier auth() {
    require(wards[msg.sender] == 1, "L2DAITokenBridge/not-authorized");
    _;
  }
```

15.4.3 constructor(_messenger) [OVM_CrossDomainEnabled] X

```
    /**
     * @param _messenger Address of the CrossDomainMessenger on the current
        ↪ layer.
     */
    constructor(
        address _messenger
    ) {
        messenger = _messenger;
    }
```

15.4.4 getCrossDomainMessenger() [OVM_CrossDomainEnabled]

```
    /**
     * Gets the messenger, usually from storage. This function is exposed in
        ↪ case a child contract
     * needs to override.
     * @return The address of the cross-domain messenger contract which should
        ↪ be used.
     */
    function getCrossDomainMessenger()
        internal
        virtual
        returns (
            iOVM_CrossDomainMessenger
        )
    {
        return iOVM_CrossDomainMessenger(messenger);
    }
```

15.4.5 sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message) [OVM_CrossDomainEnabled]

```
    /**
     * Sends a message to an account on another domain
     * @param _crossDomainTarget The intended recipient on the destination
        ↪ domain
     * @param _message The data to send to the target (usually calldata to a
        ↪ function with
     *  `onlyFromCrossDomainAccount()`
     * @param _gasLimit The gasLimit for the receipt of the message on the
        ↪ target domain.
     */
    function sendCrossDomainMessage(
        address _crossDomainTarget,
        uint32 _gasLimit,
        bytes memory _message
    )
        internal
    {
        getCrossDomainMessenger().sendMessage(_crossDomainTarget, _message,
            ↪ _gasLimit);
    }
```

15.4.6 rely(usr) X a

```
  function rely(address usr) external auth {
    wards[usr] = 1;
    emit Rely(usr);
  }
```

15.4.7 deny(usr) X a

```
  function deny(address usr) external auth {
    wards[usr] = 0;
    emit Deny(usr);
  }
```

15.4.8 constructor(_l2CrossDomainMessenger, _l2Token, _l1Token, _l1DAITokenBridge) X

```
  constructor(
    address _l2CrossDomainMessenger,
    address _l2Token,
    address _l1Token,
    address _l1DAITokenBridge
  ) public OVM_CrossDomainEnabled(_l2CrossDomainMessenger) {
    wards[msg.sender] = 1;
    emit Rely(msg.sender);

    l2Token = _l2Token;
    l1Token = _l1Token;
    l1DAITokenBridge = _l1DAITokenBridge;
  }
```

15.4.9 close() X a

```
  function close() external auth {
    isOpen = 0;

    emit Closed();
  }
```

15.4.10 withdraw(_l2Token, _amount, _l1Gas, _data) X

```
  function withdraw(
    address _l2Token,
    uint256 _amount,
    uint32 _l1Gas,
    bytes calldata _data
```

```
) external virtual override {
  require(_l2Token == l2Token, "L2DAITokenBridge/token-not-dai");

  _initiateWithdrawal(msg.sender, msg.sender, _amount, _l1Gas, _data);
}
```

15.4.11 withdrawTo(_l2Token, _to, _amount, _l1Gas, _data) X

```
function withdrawTo(
  address _l2Token,
  address _to,
  uint256 _amount,
  uint32 _l1Gas,
  bytes calldata _data
) external virtual override {
  require(_l2Token == l2Token, "L2DAITokenBridge/token-not-dai");

  _initiateWithdrawal(msg.sender, _to, _amount, _l1Gas, _data);
}
```

15.4.12 _initiateWithdrawal(_from, _to, _amount, _l1Gas, _data)

```
// When a withdrawal is initiated, we burn the withdrawer's funds to prevent
    ↪ subsequent L2 usage.
function _initiateWithdrawal(
  address _from,
  address _to,
  uint256 _amount,
  uint32 _l1Gas,
  bytes calldata _data
) internal {
  // do not allow initiaitng new xchain messages if bridge is closed
  require(isOpen == 1, "L2DAITokenBridge/closed");

  Mintable(l2Token).burn(msg.sender, _amount);

  bytes memory message = abi.encodeWithSelector(
    iOVM_L1ERC20Bridge.finalizeERC20Withdrawal.selector,
    l1Token,
    l2Token,
    _from,
    _to,
    _amount,
    _data
  );

  sendCrossDomainMessage(l1DAITokenBridge, _l1Gas, message);

  emit WithdrawalInitiated(l1Token, l2Token, msg.sender, _to, _amount, _data);
}
```

15.4.13 finalizeDeposit(_l1Token, _l2Token, _from, _to, _amount, _data) X

```
// When a deposit is finalized, we credit the account on L2 with the same
    ↪ amount of tokens.
function finalizeDeposit(
  address _l1Token,
  address _l2Token,
  address _from,
  address _to,
  uint256 _amount,
  bytes calldata _data
) external virtual override onlyFromCrossDomainAccount(l1DAITokenBridge) {
  require(_l1Token == l1Token && _l2Token == l2Token, "L2DAITokenBridge/token-
      ↪ not-dai");

  Mintable(l2Token).mint(_to, _amount);

  emit DepositFinalized(_l1Token, _l2Token, _from, _to, _amount, _data);
}
```

15.5 contract L2GovernanceRelay(2)

```
// Receive xchain message from L1 counterpart and execute given spell

contract L2GovernanceRelay is OVM_CrossDomainEnabled {
  address public immutable l1GovernanceRelay;
}
```

Inherited:

```
/**
 * @title OVM_CrossDomainEnabled
 * @dev Helper contract for contracts performing cross-domain communications
 *
 * Compiler used: defined by inheriting contract
 * Runtime target: defined by inheriting contract
 */
contract OVM_CrossDomainEnabled {

    /*************
     * Variables *
     *************/

    // Messenger contract used to send and recieve messages from the other
       ↪ domain.
    address public messenger;


    /***************
     * Constructor *
     ***************/


    /**********************
     * Function Modifiers *
     **********************/


    /**********************
     * Internal Functions *
     **********************/
}
```

15.5.1 modifier onlyFromCrossDomainAccount(_sourceDomainAccount) [OVM_CrossDomainEnabled]

```
    /**
     * Enforces that the modified function is only callable by a specific cross-
        ↪ domain account.
     * @param _sourceDomainAccount The only account on the originating domain
        ↪ which is
     *  authenticated to call this function.
     */
    modifier onlyFromCrossDomainAccount(
        address _sourceDomainAccount
    ) {
        require(
            msg.sender == address(getCrossDomainMessenger()),
            "OVM_XCHAIN: messenger contract unauthenticated"
        );

        require(
            getCrossDomainMessenger().xDomainMessageSender() ==
                ↪ _sourceDomainAccount,
            "OVM_XCHAIN: wrong sender of cross-domain message"
        );
        _;
    }
```

15.5.2 constructor(_messenger) [OVM_CrossDomainEnabled] X

```
    /**
     * @param _messenger Address of the CrossDomainMessenger on the current
        ↪ layer.
     */
    constructor(
        address _messenger
    ) {
        messenger = _messenger;
    }
```

15.5.3 getCrossDomainMessenger() [OVM_CrossDomainEnabled]

```
    /**
     * Gets the messenger, usually from storage. This function is exposed in
        ↪ case a child contract
     * needs to override.
     * @return The address of the cross-domain messenger contract which should
        ↪ be used.
     */
    function getCrossDomainMessenger()
        internal
        virtual
        returns (
            iOVM_CrossDomainMessenger
        )
    {
        return iOVM_CrossDomainMessenger(messenger);
    }
```

15.5.4 sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message) [OVM_CrossDomainEnabled]

```
    /**
     * Sends a message to an account on another domain
     * @param _crossDomainTarget The intended recipient on the destination
        ↪ domain
     * @param _message The data to send to the target (usually calldata to a
        ↪ function with
     *  `onlyFromCrossDomainAccount()`)
     * @param _gasLimit The gasLimit for the receipt of the message on the
        ↪ target domain.
     */
    function sendCrossDomainMessage(
        address _crossDomainTarget,
        uint32 _gasLimit,
        bytes memory _message
    )
        internal
    {
        getCrossDomainMessenger().sendMessage(_crossDomainTarget, _message,
            ↪ _gasLimit);
    }
```

15.5.5 constructor(_l2CrossDomainMessenger, _l1GovernanceRelay) X

```
constructor(address _l2CrossDomainMessenger, address _l1GovernanceRelay)
  OVM_CrossDomainEnabled(_l2CrossDomainMessenger)
{
  l1GovernanceRelay = _l1GovernanceRelay;
}
```

15.5.6 relay(target, targetData) X

```
/**
 * @dev Execute the call from L1.
 */
function relay(address target, bytes calldata targetData)
  external
  onlyFromCrossDomainAccount(l1GovernanceRelay)
{
  // Ensure no storage changes in the delegate call
  // Target address is trusted so this is mostly to avoid a human error
  // Note: we don't check l1GovernanceRelay because it's immutable
  address _messenger = messenger;

  bool ok;
  (ok, ) = target.delegatecall(targetData);
  require(ok, "L2GovernanceRelay/delegatecall-error");

  require(_messenger == messenger, "L2GovernanceRelay/illegal-storage-change")
     ↪ ;
}
```

15.6 contract Dai(3)

```
// Improved Dai token

contract Dai {

  // --- Auth ---
  mapping (address => uint256) public wards;

  // --- ERC20 Data ---
  string  public constant name     = "Dai Stablecoin";
  string  public constant symbol   = "DAI";
  string  public constant version  = "2";
  uint8   public constant decimals = 18;
  uint256 public totalSupply;

  mapping (address => uint256)                      public balanceOf;
  mapping (address => mapping (address => uint256)) public allowance;
  mapping (address => uint256)                      public nonces;

  event Approval(address indexed owner, address indexed spender, uint256 value);
  event Transfer(address indexed from, address indexed to, uint256 value);
  event Rely(address indexed usr);
  event Deny(address indexed usr);

  // --- EIP712 niceties ---
  uint256 public immutable deploymentChainId;
  bytes32 private immutable _DOMAIN_SEPARATOR;
  bytes32 public constant PERMIT_TYPEHASH = keccak256("Permit(address owner,
      ↪ address spender,uint256 value,uint256 nonce,uint256 deadline)");
}
```

15.6.1 modifier auth()

```
  modifier auth {
    require(wards[msg.sender] == 1, "Dai/not-authorized");
    _;
  }
```

15.6.2 rely(usr) X a

```
  function rely(address usr) external auth {
    wards[usr] = 1;
    emit Rely(usr);
  }
```

15.6.3 deny(usr) X a

```
  function deny(address usr) external auth {
    wards[usr] = 0;
    emit Deny(usr);
  }
```

15.6.4 _add(x, y)

```
  // --- Math ---
  function _add(uint256 x, uint256 y) internal pure returns (uint256 z) {
    require((z = x + y) >= x);
  }
```

15.6.5 _sub(x, y)

```
function _sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
  require((z = x - y) <= x);
}
```

15.6.6 constructor() X

```
constructor() public {
  wards[msg.sender] = 1;
  emit Rely(msg.sender);

  uint256 chainId;
  assembly {chainId := chainid()}
  deploymentChainId = chainId;
  _DOMAIN_SEPARATOR = _calculateDomainSeparator(chainId);
}
```

15.6.7 _calculateDomainSeparator(chainId)

```
function _calculateDomainSeparator(uint256 chainId) private view returns (
    ↪ bytes32) {
  return keccak256(
    abi.encode(
      keccak256("EIP712Domain(string name,string version,uint256 chainId,
          ↪ address verifyingContract)"),
      keccak256(bytes(name)),
      keccak256(bytes(version)),
      chainId,
      address(this)
    )
  );
}
```

15.6.8 DOMAIN_SEPARATOR()

```
function DOMAIN_SEPARATOR() external view returns (bytes32) {
  uint256 chainId;
  assembly {chainId := chainid()}
  return chainId == deploymentChainId ? _DOMAIN_SEPARATOR :
      ↪ _calculateDomainSeparator(chainId);
}
```

15.6.9 transfer(to, value) X

```
// --- ERC20 Mutations ---
function transfer(address to, uint256 value) external returns (bool) {
  require(to != address(0) && to != address(this), "Dai/invalid-address");
  uint256 balance = balanceOf[msg.sender];
  require(balance >= value, "Dai/insufficient-balance");

  balanceOf[msg.sender] = balance - value;
  balanceOf[to] += value;

  emit Transfer(msg.sender, to, value);

  return true;
}
```

15.6.10 transferFrom(from, to, value) X

```
function transferFrom(address from, address to, uint256 value) external
    ↪ returns (bool) {
  require(to != address(0) && to != address(this), "Dai/invalid-address");
  uint256 balance = balanceOf[from];
  require(balance >= value, "Dai/insufficient-balance");

  if (from != msg.sender) {
    uint256 allowed = allowance[from][msg.sender];
    if (allowed != type(uint256).max) {
      require(allowed >= value, "Dai/insufficient-allowance");

      allowance[from][msg.sender] = allowed - value;
    }
  }

  balanceOf[from] = balance - value;
  balanceOf[to] += value;

  emit Transfer(from, to, value);

  return true;
}
```

15.6.11 approve(spender, value) X

```
function approve(address spender, uint256 value) external returns (bool) {
  allowance[msg.sender][spender] = value;

  emit Approval(msg.sender, spender, value);

  return true;
}
```

15.6.12 increaseAllowance(spender, addedValue) X

```
function increaseAllowance(address spender, uint256 addedValue) external
    ↪ returns (bool) {
  uint256 newValue = _add(allowance[msg.sender][spender], addedValue);
  allowance[msg.sender][spender] = newValue;

  emit Approval(msg.sender, spender, newValue);

  return true;
}
```

15.6.13 decreaseAllowance(spender, subtractedValue) X

```
function decreaseAllowance(address spender, uint256 subtractedValue) external
    ↪ returns (bool) {
  uint256 allowed = allowance[msg.sender][spender];
  require(allowed >= subtractedValue, "Dai/insufficient-allowance");
  allowed = allowed - subtractedValue;
  allowance[msg.sender][spender] = allowed;

  emit Approval(msg.sender, spender, allowed);

  return true;
}
```

15.6.14 mint(to, value) X a

```
// --- Mint/Burn ---
function mint(address to, uint256 value) external auth {
  require(to != address(0) && to != address(this), "Dai/invalid-address");
  balanceOf[to] = balanceOf[to] + value; // note: we don't need an overflow
      ↪ check here b/c balanceOf[to] <= totalSupply and there is an overflow
      ↪ check below
  totalSupply   = _add(totalSupply, value);

  emit Transfer(address(0), to, value);
}
```

15.6.15 burn(from, value) X

```
function burn(address from, uint256 value) external {
  uint256 balance = balanceOf[from];
  require(balance >= value, "Dai/insufficient-balance");

  if (from != msg.sender && wards[msg.sender] != 1) {
    uint256 allowed = allowance[from][msg.sender];
    if (allowed != type(uint256).max) {
      require(allowed >= value, "Dai/insufficient-allowance");

      allowance[from][msg.sender] = allowed - value;
    }
  }

  balanceOf[from] = balance - value; // note: we don't need overflow checks b/
      ↪ c require(balance >= value) and balance <= totalSupply
  totalSupply     = totalSupply - value;

  emit Transfer(from, address(0), value);
}
```

15.6.16 permit(owner, spender, value, deadline, v, r, s) X

```
// --- Approve by signature ---
function permit(address owner, address spender, uint256 value, uint256
    ↪ deadline, uint8 v, bytes32 r, bytes32 s) external {
  require(block.timestamp <= deadline, "Dai/permit-expired");

  uint256 chainId;
  assembly {chainId := chainid()}

  bytes32 digest =
    keccak256(abi.encodePacked(
        "\x19\x01",
        chainId == deploymentChainId ? _DOMAIN_SEPARATOR :
            ↪ _calculateDomainSeparator(chainId),
        keccak256(abi.encode(
          PERMIT_TYPEHASH,
          owner,
          spender,
          value,
          nonces[owner]++,
          deadline
        ))
    ));

  require(owner != address(0) && owner == ecrecover(digest, v, r, s), "Dai/
      ↪ invalid-permit");

  allowance[owner][spender] = value;
  emit Approval(owner, spender, value);
}
```

Chapter 16

StarkNet bridge

16.1 contract L1DAIBridge

```
contract L1DAIBridge {
    // --- Auth ---
    mapping(address => uint256) public wards;

    event Rely(address indexed usr);
    event Deny(address indexed usr);


    uint256 public isOpen = 1;

    event Closed();

    address public immutable starkNet;
    address public immutable dai;
    uint256 public immutable l2Dai;
    address public immutable escrow;
    uint256 public immutable l2DaiBridge;

    uint256 public ceiling = 0;
    uint256 public maxDeposit = type(uint256).max;

    uint256 constant HANDLE_WITHDRAW = 0;

    // src/starkware/cairo/lang/cairo_constants.py
    //  2 ** 251 + 17 * 2 ** 192 + 1;
    uint256 constant SN_PRIME =
        3618502788666131213697322783095070105623107215331596699973092056135872020481;
        ↪

    //  from starkware.starknet.compiler.compile import get_selector_from_name
    //  print(get_selector_from_name('handle_deposit'))
    uint256 constant DEPOSIT =
        1285101517810983806491589552491143496277809242732141897358598292095611420389;
        ↪

    //  print(get_selector_from_name('handle_force_withdrawal'))
    uint256 constant FORCE_WITHDRAW =
        1137729855293860737061629600728503767337326808607526258057644140918272132445;
        ↪

    event LogCeiling(uint256 ceiling);
    event LogMaxDeposit(uint256 maxDeposit);
    event LogDeposit(address indexed l1Sender, uint256 amount, uint256
       ↪ l2Recipient);
    event LogWithdrawal(address indexed l1Recipient, uint256 amount);
    event LogForceWithdrawal(address indexed l1Recipient, uint256 amount,
       ↪ uint256 indexed l2Sender);
    event LogStartDepositCancellation(uint256 indexed l2Receipient, uint256
       ↪ amount, uint256 nonce);
    event LogCancelDeposit(
        uint256 indexed l2Recipient, address l1Recipient, uint256 amount,
            ↪ uint256 nonce
    );
}
```

16.1.1 modifier auth()

```
    modifier auth() {
        require(wards[msg.sender] == 1, "L1DAIBridge/not-authorized");
        _;
    }
```

16.1.2 modifier whenOpen()

```
    modifier whenOpen() {
        require(isOpen == 1, "L1DAIBridge/closed");
        _;
    }
```

16.1.3 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

16.1.4 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

16.1.5 close() X a

```
    function close() external auth {
        isOpen = 0;
        emit Closed();
    }
```

16.1.6 constructor(_starkNet, _dai, _l2Dai, _escrow, _l2DaiBridge) X

```
    constructor(
        address _starkNet,
        address _dai,
        uint256 _l2Dai,
        address _escrow,
        uint256 _l2DaiBridge
    ) {
        wards[msg.sender] = 1;
        emit Rely(msg.sender);

        starkNet = _starkNet;
        dai = _dai;
        l2Dai = _l2Dai;
        escrow = _escrow;
        l2DaiBridge = _l2DaiBridge;
    }
```

16.1.7 setCeiling(_ceiling) X a

```
function setCeiling(uint256 _ceiling) external auth whenOpen {
    ceiling = _ceiling;
    emit LogCeiling(_ceiling);
}
```

16.1.8 setMaxDeposit(_maxDeposit) X a

```
function setMaxDeposit(uint256 _maxDeposit) external auth whenOpen {
    maxDeposit = _maxDeposit;
    emit LogMaxDeposit(_maxDeposit);
}
```

16.1.9 deposit(amount, l2Recipient) X

```
// slither-disable-next-line similar-names
function deposit(
    uint256 amount,
    uint256 l2Recipient
) external payable whenOpen {
    emit LogDeposit(msg.sender, amount, l2Recipient);

    require(l2Recipient != 0 && l2Recipient != l2Dai && l2Recipient <
        ↪ SN_PRIME, "L1DAIBridge/invalid-address");

    require(amount <= maxDeposit, "L1DAIBridge/above-max-deposit");

    TokenLike(dai).transferFrom(msg.sender, escrow, amount);

    require(
        TokenLike(dai).balanceOf(escrow) <= ceiling,
        "L1DAIBridge/above-ceiling"
    );

    uint256[] memory payload = new uint256[](4);
    payload[0] = l2Recipient;
    (payload[1], payload[2]) = toSplitUint(amount);
    payload[3] = uint256(uint160(msg.sender));

    StarkNetLike(starkNet).sendMessageToL2{value: msg.value}(l2DaiBridge,
        ↪ DEPOSIT, payload);
}
```

16.1.10 toSplitUint(value)

```
function toSplitUint(uint256 value) internal pure returns (uint256, uint256)
    ↪ {
  uint256 low = value & ((1 << 128) - 1);
  uint256 high = value >> 128;
  return (low, high);
}
```

16.1.11 withdraw(amount, l1Recipient) X

```
// slither-disable-next-line similar-names
function withdraw(uint256 amount, address l1Recipient) external {
    emit LogWithdrawal(l1Recipient, amount);

    uint256[] memory payload = new uint256[](4);
    payload[0] = HANDLE_WITHDRAW;
    payload[1] = uint256(uint160(msg.sender));
```

```
        (payload[2], payload[3]) = toSplitUint(amount);

        StarkNetLike(starkNet).consumeMessageFromL2(l2DaiBridge, payload);
        TokenLike(dai).transferFrom(escrow, l1Recipient, amount);
    }
```

16.1.12 forceWithdrawal(amount, l2Sender) X

```
    function forceWithdrawal(uint256 amount, uint256 l2Sender) external payable
       ↪ whenOpen {
        emit LogForceWithdrawal(msg.sender, amount, l2Sender);

        uint256[] memory payload = new uint256[](4);
        payload[0] = l2Sender;
        payload[1] = uint256(uint160(msg.sender));
        (payload[2], payload[3]) = toSplitUint(amount);

        StarkNetLike(starkNet).sendMessageToL2{value: msg.value}(l2DaiBridge,
           ↪ FORCE_WITHDRAW, payload);
    }
```

16.1.13 startDepositCancellation(amount, l2Recipient, nonce) X

```
    function startDepositCancellation(
        uint256 amount,
        uint256 l2Recipient,
        uint256 nonce
    ) external {
        emit LogStartDepositCancellation(l2Recipient, amount, nonce);

        uint256[] memory payload = new uint256[](4);
        payload[0] = l2Recipient;
        (payload[1], payload[2]) = toSplitUint(amount);
        payload[3] = uint256(uint160(msg.sender));

        StarkNetLike(starkNet).startL1ToL2MessageCancellation(l2DaiBridge,
           ↪ DEPOSIT, payload, nonce);
    }
```

16.1.14 cancelDeposit(amount, l2Recipient, l1Recipient, nonce) X

```
    function cancelDeposit(
        uint256 amount,
        uint256 l2Recipient,
        // slither-disable-next-line similar-names
        address l1Recipient,
        uint256 nonce
    ) external {
        emit LogCancelDeposit(l2Recipient, l1Recipient, amount, nonce);

        uint256[] memory payload = new uint256[](4);
        payload[0] = l2Recipient;
        (payload[1], payload[2]) = toSplitUint(amount);
        payload[3] = uint256(uint160(msg.sender));

        StarkNetLike(starkNet).cancelL1ToL2Message(l2DaiBridge, DEPOSIT, payload
           ↪ , nonce);
        TokenLike(dai).transferFrom(escrow, l1Recipient, amount);
    }
```

16.2 contract L1GovernanceRelay(3)

```
contract L1GovernanceRelay {
  // --- Auth ---
  mapping(address => uint256) public wards;

  event Rely(address indexed usr);
  event Deny(address indexed usr);

  uint256 constant RELAY_SELECTOR =
      ↪ 300224956480472355485152391090755024345070441743081995053718200325371913697;
      ↪

  address public immutable starkNet;
  uint256 public immutable l2GovernanceRelay;

}
```

16.2.1 modifier auth()

```
  modifier auth() {
    require(wards[msg.sender] == 1, "L1GovernanceRelay/not-authorized");
    _;
  }
```

16.2.2 rely(usr) X a

```
  function rely(address usr) external auth {
    wards[usr] = 1;
    emit Rely(usr);
  }
```

16.2.3 deny(usr) X a

```
  function deny(address usr) external auth {
    wards[usr] = 0;
    emit Deny(usr);
  }
```

16.2.4 constructor(_starkNet, _l2GovernanceRelay) X

```
  constructor(address _starkNet, uint256 _l2GovernanceRelay) {
    wards[msg.sender] = 1;
    emit Rely(msg.sender);

    starkNet = _starkNet;
    l2GovernanceRelay = _l2GovernanceRelay;
  }
```

16.2.5 relay(spell) X a

```
  // Forward a call to be repeated on L2
  function relay(uint256 spell) external payable auth {
    uint256[] memory payload = new uint256[](1);
    payload[0] = spell;
    StarkNetLike(starkNet).sendMessageToL2{value: msg.value}(l2GovernanceRelay,
        ↪ RELAY_SELECTOR, payload);
  }
```

16.3 contract L1DAITeleportGateway

```
contract L1DAITeleportGateway {
  address public immutable starkNet;
  address public immutable l1Token;
  uint256 public immutable l2TeleportGateway;
  address public immutable l1Escrow;
  TeleportRouter public immutable l1TeleportRouter;

  uint256 constant HANDLE_REGISTER_TELEPORT = 0;
  uint256 constant HANDLE_FLUSH = 1;
}
```

16.3.1 constructor(_starkNet, _l1Token, _l2TeleportGateway, _l1Escrow, _l1TeleportRouter) X

```
  constructor(
    address _starkNet,
    address _l1Token,
    uint256 _l2TeleportGateway,
    address _l1Escrow,
    address _l1TeleportRouter
  ) {

    starkNet = _starkNet;
    l1Token = _l1Token;
    l2TeleportGateway = _l2TeleportGateway;
    l1Escrow = _l1Escrow;
    l1TeleportRouter = TeleportRouter(_l1TeleportRouter);
    // Approve the router to pull DAI from this contract during settle() (after
       ↪ the DAI has been pulled by this contract from the escrow)
    ApproveLike(_l1Token).approve(_l1TeleportRouter, type(uint256).max);
  }
```

16.3.2 finalizeFlush(targetDomain, daiToFlush) X

```
  function finalizeFlush(bytes32 targetDomain, uint256 daiToFlush)
    external
  {
    uint256[] memory payload = new uint256[](4);
    payload[0] = HANDLE_FLUSH;
    payload[1] = toL2String(targetDomain);
    (payload[2], payload[3]) = toSplitUint(daiToFlush);

    StarkNetLike(starkNet).consumeMessageFromL2(l2TeleportGateway, payload);

    // Pull DAI from the escrow to this contract
    TokenLike(l1Token).transferFrom(l1Escrow, address(this), daiToFlush);
    // The router will pull the DAI from this contract
    l1TeleportRouter.settle(targetDomain, daiToFlush);
  }
```

16.3.3 finalizeRegisterTeleport(teleport) X

```
  function finalizeRegisterTeleport(TeleportGUID calldata teleport)
    external
  {
    uint256[] memory payload = new uint256[](8);
    payload[0] = HANDLE_REGISTER_TELEPORT;
    payload[1] = toL2String(teleport.sourceDomain); // bytes32 -> uint256
    payload[2] = toL2String(teleport.targetDomain); // bytes32 -> uint256
    payload[3] = uint256(teleport.receiver); // bytes32 -> uint256
    payload[4] = uint256(teleport.operator); // bytes32 -> uint256
    payload[5] = uint256(teleport.amount); // uint128 -> uint256
    payload[6] = uint256(teleport.nonce); // uint80 -> uint256
    payload[7] = uint256(teleport.timestamp); // uint48 -> uint256

    StarkNetLike(starkNet).consumeMessageFromL2(l2TeleportGateway, payload);

    l1TeleportRouter.requestMint(teleport, 0, 0);
  }
```

16.3.4 toL2String(str)

```
  function toL2String(bytes32 str) internal pure returns (uint256) {
    while (str[31] == '\x00') {
      str = str >> 8;
    }
    return uint256(str);
  }
```

16.3.5 toSplitUint(value)

```
  function toSplitUint(uint256 value) internal pure returns (uint256, uint256) {
    uint256 low = value & ((1 << 128) - 1);
    uint256 high = value >> 128;
    return (low, high);
  }
```

16.4 contract DAIMock

```
contract DAIMock is ERC20 {
}
```

Inherited:

```
/**
 * @dev Implementation of the {IERC20} interface.
 *
 * This implementation is agnostic to the way tokens are created. This means
 * that a supply mechanism has to be added in a derived contract using {_mint}.
 * For a generic mechanism see {ERC20PresetMinterPauser}.
 *
 * TIP: For a detailed writeup see our guide
 * https://forum.zeppelin.solutions/t/how-to-implement-erc20-supply-mechanisms
    ↪ /226[How
 * to implement supply mechanisms].
 *
 * We have followed general OpenZeppelin Contracts guidelines: functions revert
 * instead returning `false` on failure. This behavior is nonetheless
 * conventional and does not conflict with the expectations of ERC20
 * applications.
 *
 * Additionally, an {Approval} event is emitted on calls to {transferFrom}.
 * This allows applications to reconstruct the allowance for all accounts just
 * by listening to said events. Other implementations of the EIP may not emit
 * these events, as it isn't required by the specification.
 *
 * Finally, the non-standard {decreaseAllowance} and {increaseAllowance}
 * functions have been added to mitigate the well-known issues around setting
 * allowances. See {IERC20-approve}.
 */
contract ERC20 is Context, IERC20, IERC20Metadata {
    mapping(address => uint256) private _balances;

    mapping(address => mapping(address => uint256)) private _allowances;

    uint256 private _totalSupply;

    string private _name;
    string private _symbol;

    /**
     * @dev Hook that is called before any transfer of tokens. This includes
     * minting and burning.
     *
     * Calling conditions:
     *
     * - when `from` and `to` are both non-zero, `amount` of ``from``'s tokens
     * will be transferred to `to`.
     * - when `from` is zero, `amount` tokens will be minted for `to`.
     * - when `to` is zero, `amount` of ``from``'s tokens will be burned.
     * - `from` and `to` are never both zero.
     *
     * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#
        ↪ using-hooks[Using Hooks].
     */
    function _beforeTokenTransfer(
        address from,
        address to,
        uint256 amount
    ) internal virtual {}

    /**
     * @dev Hook that is called after any transfer of tokens. This includes
     * minting and burning.
     *
     * Calling conditions:
```

```
     *
     * - when `from` and `to` are both non-zero, `amount` of ``from``'s tokens
     * has been transferred to `to`.
     * - when `from` is zero, `amount` tokens have been minted for `to`.
     * - when `to` is zero, `amount` of ``from``'s tokens have been burned.
     * - `from` and `to` are never both zero.
     *
     * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#
        ↪ using-hooks[Using Hooks].
     */
    function _afterTokenTransfer(
        address from,
        address to,
        uint256 amount
    ) internal virtual {}
}
```

```
/**
 * @dev Provides information about the current execution context, including the
 * sender of the transaction and its data. While these are generally available
 * via msg.sender and msg.data, they should not be accessed in such a direct
 * manner, since when dealing with meta-transactions the account sending and
 * paying for execution may not be the actual sender (as far as an application
 * is concerned).
 *
 * This contract is only required for intermediate, library-like contracts.
 */
abstract contract Context {
}
```

16.4.1 constructor(name_, symbol_) [ERC20] X

```
    /**
     * @dev Sets the values for {name} and {symbol}.
     *
     * The default value of {decimals} is 18. To select a different value for
     * {decimals} you should overload it.
     *
     * All two of these values are immutable: they can only be set once during
     * construction.
     */
    constructor(string memory name_, string memory symbol_) {
        _name = name_;
        _symbol = symbol_;
    }
```

16.4.2 name() [ERC20]

```
    /**
     * @dev Returns the name of the token.
     */
    function name() public view virtual override returns (string memory) {
        return _name;
    }
```

16.4.3 symbol() [ERC20]

```
    /**
     * @dev Returns the symbol of the token, usually a shorter version of the
     * name.
     */
    function symbol() public view virtual override returns (string memory) {
        return _symbol;
    }
```

16.4.4 decimals() [ERC20]

```
/**
 * @dev Returns the number of decimals used to get its user representation.
 * For example, if `decimals` equals `2`, a balance of `505` tokens should
 * be displayed to a user as `5.05` (`505 / 10 ** 2`).
 *
 * Tokens usually opt for a value of 18, imitating the relationship between
 * Ether and Wei. This is the value {ERC20} uses, unless this function is
 * overridden;
 *
 * NOTE: This information is only used for _display_ purposes: it in
 * no way affects any of the arithmetic of the contract, including
 * {IERC20-balanceOf} and {IERC20-transfer}.
 */
function decimals() public view virtual override returns (uint8) {
    return 18;
}
```

16.4.5 totalSupply() [ERC20]

```
/**
 * @dev See {IERC20-totalSupply}.
 */
function totalSupply() public view virtual override returns (uint256) {
    return _totalSupply;
}
```

16.4.6 balanceOf(account) [ERC20]

```
/**
 * @dev See {IERC20-balanceOf}.
 */
function balanceOf(address account) public view virtual override returns (
    ↪ uint256) {
    return _balances[account];
}
```

16.4.7 transfer(to, amount) [ERC20] X

```
/**
 * @dev See {IERC20-transfer}.
 *
 * Requirements:
 *
 * - `to` cannot be the zero address.
 * - the caller must have a balance of at least `amount`.
 */
function transfer(address to, uint256 amount) public virtual override
    ↪ returns (bool) {
    address owner = _msgSender();
    _transfer(owner, to, amount);
    return true;
}
```

16.4.8 allowance(owner, spender) [ERC20]

```
/**
 * @dev See {IERC20-allowance}.
 */
function allowance(address owner, address spender) public view virtual
    ↪ override returns (uint256) {
        return _allowances[owner][spender];
    }
```

16.4.9 approve(spender, amount) [ERC20] X

```
    /**
     * @dev See {IERC20-approve}.
     *
     * NOTE: If `amount` is the maximum `uint256`, the allowance is not updated
        ↪ on
     * `transferFrom`. This is semantically equivalent to an infinite approval.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     */
    function approve(address spender, uint256 amount) public virtual override
       ↪ returns (bool) {
        address owner = _msgSender();
        _approve(owner, spender, amount);
        return true;
    }
```

16.4.10 transferFrom(from, to, amount) [ERC20] X

```
    /**
     * @dev See {IERC20-transferFrom}.
     *
     * Emits an {Approval} event indicating the updated allowance. This is not
     * required by the EIP. See the note at the beginning of {ERC20}.
     *
     * NOTE: Does not update the allowance if the current allowance
     * is the maximum `uint256`.
     *
     * Requirements:
     *
     * - `from` and `to` cannot be the zero address.
     * - `from` must have a balance of at least `amount`.
     * - the caller must have allowance for ``from``'s tokens of at least
     * `amount`.
     */
    function transferFrom(
        address from,
        address to,
        uint256 amount
    ) public virtual override returns (bool) {
        address spender = _msgSender();
        _spendAllowance(from, spender, amount);
        _transfer(from, to, amount);
        return true;
    }
```

16.4.11 increaseAllowance(spender, addedValue) [ERC20] X

```
    /**
     * @dev Atomically increases the allowance granted to `spender` by the
        ↪ caller.
     *
     * This is an alternative to {approve} that can be used as a mitigation for
     * problems described in {IERC20-approve}.
     *
     * Emits an {Approval} event indicating the updated allowance.
     *
     * Requirements:
```

```
     *
     * - `spender` cannot be the zero address.
     */
    function increaseAllowance(address spender, uint256 addedValue) public
        ↪ virtual returns (bool) {
        address owner = _msgSender();
        _approve(owner, spender, allowance(owner, spender) + addedValue);
        return true;
    }
```

16.4.12 decreaseAllowance(spender, subtractedValue) [ERC20] X

```
    /**
     * @dev Atomically decreases the allowance granted to `spender` by the
         ↪ caller.
     *
     * This is an alternative to {approve} that can be used as a mitigation for
     * problems described in {IERC20-approve}.
     *
     * Emits an {Approval} event indicating the updated allowance.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     * - `spender` must have allowance for the caller of at least
     * `subtractedValue`.
     */
    function decreaseAllowance(address spender, uint256 subtractedValue) public
        ↪ virtual returns (bool) {
        address owner = _msgSender();
        uint256 currentAllowance = allowance(owner, spender);
        require(currentAllowance >= subtractedValue, "ERC20: decreased allowance
            ↪  below zero");
        unchecked {
            _approve(owner, spender, currentAllowance - subtractedValue);
        }

        return true;
    }
```

16.4.13 _transfer(from, to, amount) [ERC20]

```
    /**
     * @dev Moves `amount` of tokens from `sender` to `recipient`.
     *
     * This internal function is equivalent to {transfer}, and can be used to
     * e.g. implement automatic token fees, slashing mechanisms, etc.
     *
     * Emits a {Transfer} event.
     *
     * Requirements:
     *
     * - `from` cannot be the zero address.
     * - `to` cannot be the zero address.
     * - `from` must have a balance of at least `amount`.
     */
    function _transfer(
        address from,
        address to,
        uint256 amount
    ) internal virtual {
        require(from != address(0), "ERC20: transfer from the zero address");
        require(to != address(0), "ERC20: transfer to the zero address");

        _beforeTokenTransfer(from, to, amount);
        uint256 fromBalance = _balances[from];
        require(fromBalance >= amount, "ERC20: transfer amount exceeds balance")
            ↪ ;
        unchecked {
            _balances[from] = fromBalance - amount;
        }
        _balances[to] += amount;

        emit Transfer(from, to, amount);

        _afterTokenTransfer(from, to, amount);
    }
```

16.4.14 _mint(account, amount) [ERC20]

```
    /** @dev Creates `amount` tokens and assigns them to `account`, increasing
     * the total supply.
     *
     * Emits a {Transfer} event with `from` set to the zero address.
     *
     * Requirements:
     *
     * - `account` cannot be the zero address.
     */
    function _mint(address account, uint256 amount) internal virtual {
        require(account != address(0), "ERC20: mint to the zero address");

        _beforeTokenTransfer(address(0), account, amount);

        _totalSupply += amount;
        _balances[account] += amount;
        emit Transfer(address(0), account, amount);

        _afterTokenTransfer(address(0), account, amount);
    }
```

16.4.15 _burn(account, amount) [ERC20]

```
    /**
     * @dev Destroys `amount` tokens from `account`, reducing the
     * total supply.
     *
     * Emits a {Transfer} event with `to` set to the zero address.
     *
     * Requirements:
     *
     * - `account` cannot be the zero address.
     * - `account` must have at least `amount` tokens.
     */
    function _burn(address account, uint256 amount) internal virtual {
        require(account != address(0), "ERC20: burn from the zero address");

        _beforeTokenTransfer(account, address(0), amount);

        uint256 accountBalance = _balances[account];
        require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
        unchecked {
            _balances[account] = accountBalance - amount;
        }
        _totalSupply -= amount;

        emit Transfer(account, address(0), amount);

        _afterTokenTransfer(account, address(0), amount);
    }
```

16.4.16 _approve(owner, spender, amount) [ERC20]

```
    /**
     * @dev Sets `amount` as the allowance of `spender` over the `owner` s
        ↪ tokens.
     *
     * This internal function is equivalent to `approve`, and can be used to
     * e.g. set automatic allowances for certain subsystems, etc.
     *
     * Emits an {Approval} event.
     *
     * Requirements:
     *
     * - `owner` cannot be the zero address.
     * - `spender` cannot be the zero address.
     */
    function _approve(
        address owner,
        address spender,
        uint256 amount
    ) internal virtual {
        require(owner != address(0), "ERC20: approve from the zero address");
        require(spender != address(0), "ERC20: approve to the zero address");

        _allowances[owner][spender] = amount;
        emit Approval(owner, spender, amount);
    }
```

16.4.17 _spendAllowance(owner, spender, amount) [ERC20]

```
    /**
     * @dev Updates `owner` s allowance for `spender` based on spent `amount`.
     *
     * Does not update the allowance amount in case of infinite allowance.
     * Revert if not enough allowance is available.
     *
     * Might emit an {Approval} event.
     */
    function _spendAllowance(
        address owner,
        address spender,
        uint256 amount
    ) internal virtual {
        uint256 currentAllowance = allowance(owner, spender);
        if (currentAllowance != type(uint256).max) {
            require(currentAllowance >= amount, "ERC20: insufficient allowance")
                ↪ ;
            unchecked {
                _approve(owner, spender, currentAllowance - amount);
            }
        }
    }
```

16.4.18 _msgSender() [Context]

```
    function _msgSender() internal view virtual returns (address) {
        return msg.sender;
    }
```

16.4.19 _msgData() [Context]

```
    function _msgData() internal view virtual returns (bytes calldata) {
        return msg.data;
    }
```

16.4.20 constructor() X

```
    constructor () ERC20('DAI', 'DAI') {
        _mint(msg.sender, 1_000_000 * 1 ether);
    }
```

16.5 contract TeleportRouterMock

```
// slither-disable-next-line missing-inheritance
contract TeleportRouterMock {

  event RequestMint(TeleportGUID teleportGUID, uint256 maxFeePercentage, uint256
      ↪  operatorFee);
  event Settle(bytes32 targetDomain, uint256 batchedDaiToFlush);
}
```

16.5.1 requestMint(teleportGUID, maxFeePercentage, operatorFee) X

```
  function requestMint(
    TeleportGUID calldata teleportGUID,
    uint256 maxFeePercentage,
    uint256 operatorFee
  ) external returns (uint256 postFeeAmount, uint256 totalFee) {
    emit RequestMint(teleportGUID, maxFeePercentage, operatorFee);
    postFeeAmount = maxFeePercentage;
    totalFee = operatorFee;
  }
```

16.5.2 settle(targetDomain, batchedDaiToFlush) X

```
  function settle(bytes32 targetDomain, uint256 batchedDaiToFlush) external {
    emit Settle(targetDomain, batchedDaiToFlush);
  }
```

16.6 contract L1Escrow(3)

```
// Escrow funds on L1, manage approval rights
contract L1Escrow {

  // --- Auth ---
  mapping (address => uint256) public wards;

  event Rely(address indexed usr);
  event Deny(address indexed usr);

  event Approve(address indexed token, address indexed spender, uint256 value);
}
```

16.6.1 modifier auth()

```
  modifier auth {
    require(wards[msg.sender] == 1, "L1Escrow/not-authorized");
    _;
  }
```

16.6.2 rely(usr) X a

```
  function rely(address usr) external auth {
    wards[usr] = 1;
    emit Rely(usr);
  }
```

16.6.3 deny(usr) X a

```
  function deny(address usr) external auth {
    wards[usr] = 0;
    emit Deny(usr);
  }
```

16.6.4 constructor() X

```
  constructor() {
    wards[msg.sender] = 1;
    emit Rely(msg.sender);
  }
```

16.6.5 approve(token, spender, value) X a

```
  function approve(address token, address spender, uint256 value) external auth
     ↪ {
    emit Approve(token, spender, value);
    ApproveLike(token).approve(spender, value);
  }
```

16.7 contract L1EscrowMom

```
// Bypass governance delay to disable a L1Escrow
contract L1EscrowMom {
    address public owner;
    address public authority;

    address public immutable escrow;
    address public immutable token;

    event SetOwner(address indexed oldOwner, address indexed newOwner);
    event SetAuthority(address indexed oldAuthority, address indexed
        ↪ newAuthority);
    event Refuse(address indexed escrow, address token, address spender);
}
```

16.7.1 modifier onlyOwner()

```
    modifier onlyOwner {
        require(msg.sender == owner, "L1EscrowMom/only-owner");
        _;
    }
```

16.7.2 modifier auth()

```
    modifier auth {
        require(isAuthorized(msg.sender, msg.sig), "L1EscrowMom/not-authorized")
            ↪ ;
        _;
    }
```

16.7.3 constructor(escrow_, token_) X

```
    constructor(address escrow_, address token_) {
        owner = msg.sender;
        escrow = escrow_;
        token = token_;
        emit SetOwner(address(0), msg.sender);
    }
```

16.7.4 isAuthorized(src, sig)

```
    function isAuthorized(address src, bytes4 sig) internal view returns (bool)
        ↪ {
        if (src == address(this)) {
            return true;
        } else if (src == owner) {
            return true;
        } else if (authority == address(0)) {
            return false;
        } else {
            return AuthorityLike(authority).canCall(src, address(this), sig);
        }
    }
```

16.7.5 setOwner(owner_) X

```
    // Governance actions with delay
    function setOwner(address owner_) external onlyOwner {
        emit SetOwner(owner, owner_);
        owner = owner_;
    }
```

16.7.6 setAuthority(authority_) X

```
    function setAuthority(address authority_) external onlyOwner {
        emit SetAuthority(authority, authority_);
        authority = authority_;
    }
```

16.7.7 refuse(spender) X a

```
    // Governance action without delay
    function refuse(address spender) external auth {
        emit Refuse(escrow, token, spender);
        EscrowLike(escrow).approve(token, spender, 0);
    }
```

Chapter 17

Teleport

17.1 contract TeleportConstantFee

```
contract TeleportConstantFee is TeleportFees {
    uint256 immutable public fee;
    uint256 immutable public ttl;
}
```

17.1.1 constructor(_fee, _ttl) X

```
    /**
    * @param _fee Constant fee in WAD
    * @param _ttl Time in seconds to finalize flush (not teleport)
    **/
    constructor(uint256 _fee, uint256 _ttl) {
        fee = _fee;
        ttl = _ttl;
    }
```

17.1.2 getFee(guid, , , , amtToTake)

```
    function getFee(TeleportGUID calldata guid, uint256, int256, uint256,
        ↪ uint256 amtToTake) override external view returns (uint256) {
        // is slow withdrawal?
        if (block.timestamp >= uint256(guid.timestamp) + ttl) {
            return 0;
        }

        // is empty teleport?
        if (guid.amount == 0) {
            return 0;
        }

        return fee * amtToTake / guid.amount;
    }
```

17.2 contract TeleportJoin

```
// Primary control for extending Teleport credit
contract TeleportJoin {
    mapping (address =>        uint256) public wards;     // Auth
    mapping (bytes32 =>        address) public fees;      // Fees contract per
        ↪ source domain
    mapping (bytes32 =>        uint256) public line;      // Debt ceiling per
        ↪ source domain
    mapping (bytes32 =>         int256) public debt;      // Outstanding debt
        ↪ per source domain (can be < 0 when settlement occurs before mint)
    mapping (bytes32 => TeleportStatus) public teleports; // Approved teleports
        ↪ and pending unpaid

    address public vow;

    uint256 internal art; // We need to preserve the last art value before the
        ↪ position being skimmed (End)

    VatLike      immutable public vat;
    DaiJoinLike immutable public daiJoin;
    bytes32      immutable public ilk;
    bytes32      immutable public domain;

    uint256 constant public WAD = 10 ** 18;
    uint256 constant public RAY = 10 ** 27;

    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event File(bytes32 indexed what, address data);
    event File(bytes32 indexed what, bytes32 indexed domain, address data);
    event File(bytes32 indexed what, bytes32 indexed domain, uint256 data);
    event Register(bytes32 indexed hashGUID, TeleportGUID teleportGUID);
    event Mint(
        bytes32 indexed hashGUID, TeleportGUID teleportGUID, uint256 amount,
            ↪ uint256 maxFeePercentage, uint256 operatorFee, address originator
    );
    event Settle(bytes32 indexed sourceDomain, uint256 batchedDaiToFlush);
}
```

17.2.1 struct TeleportJoin.TeleportStatus

```
    struct TeleportStatus {
        bool    blessed;
        uint248 pending;
    }
```

17.2.2 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "TeleportJoin/not-authorized");
        _;
    }
```

17.2.3 constructor(vat_, daiJoin_, ilk_, domain_) X

```
    constructor(address vat_, address daiJoin_, bytes32 ilk_, bytes32 domain_) {
        wards[msg.sender] = 1;
        emit Rely(msg.sender);
        vat = VatLike(vat_);
        daiJoin = DaiJoinLike(daiJoin_);
        vat.hope(daiJoin_);
        daiJoin.dai().approve(daiJoin_, type(uint256).max);
        ilk = ilk_;
        domain = domain_;
    }
```

17.2.4 _min(x, y)

```
    function _min(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = x <= y ? x : y;
    }
```

17.2.5 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

17.2.6 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

17.2.7 file(what, data) X a

```
    function file(bytes32 what, address data) external auth {
        if (what == "vow") {
            vow = data;
        } else {
            revert("TeleportJoin/file-unrecognized-param");
        }
        emit File(what, data);
    }
```

17.2.8 file(what, domain_, data) X a

```
    function file(bytes32 what, bytes32 domain_, address data) external auth {
        if (what == "fees") {
            fees[domain_] = data;
        } else {
            revert("TeleportJoin/file-unrecognized-param");
        }
        emit File(what, domain_, data);
    }
```

17.2.9 file(what, domain_, data) X a

```
    function file(bytes32 what, bytes32 domain_, uint256 data) external auth {
        if (what == "line") {
            require(data <= uint256(type(int256).max), "TeleportJoin/not-allowed
              ↪ -bigger-int256");
            line[domain_] = data;
        } else {
            revert("TeleportJoin/file-unrecognized-param");
        }
        emit File(what, domain_, data);
    }
```

17.2.10 cure()

```
/**
* @dev External view function to get the total debt used by this contract [
    ↪ RAD]
**/
function cure() external view returns (uint256 cure_) {
    cure_ = art * RAY;
}
```

17.2.11 _mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee)

```
/**
* @dev Internal function that executes the mint after a teleport is
    ↪ registered
* @param teleportGUID Struct which contains the whole teleport data
* @param hashGUID Hash of the prev struct
* @param maxFeePercentage Max percentage of the withdrawn amount (in WAD) to
    ↪  be paid as fee (e.g 1% = 0.01 * WAD)
* @param operatorFee The amount of DAI to pay to the operator
* @return postFeeAmount The amount of DAI sent to the receiver after taking
    ↪ out fees
* @return totalFee The total amount of DAI charged as fees
**/
function _mint(
    TeleportGUID calldata teleportGUID,
    bytes32 hashGUID,
    uint256 maxFeePercentage,
    uint256 operatorFee
) internal returns (uint256 postFeeAmount, uint256 totalFee) {
    require(teleportGUID.targetDomain == domain, "TeleportJoin/incorrect-
        ↪ domain");

    bool vatLive = vat.live() == 1;

    uint256 line_ = vatLive ? line[teleportGUID.sourceDomain] : 0;

    int256 debt_ = debt[teleportGUID.sourceDomain];

    // Stop execution if there isn't anything available to withdraw
    uint248 pending = teleports[hashGUID].pending;
    if (int256(line_) <= debt_ || pending == 0) {
        emit Mint(hashGUID, teleportGUID, 0, maxFeePercentage, operatorFee,
            ↪ msg.sender);
        return (0, 0);
    }

    uint256 amtToTake = _min(
                            pending,
                            uint256(int256(line_) - debt_)
                        );

    uint256 fee = vatLive ? FeesLike(fees[teleportGUID.sourceDomain]).getFee
        ↪ (teleportGUID, line_, debt_, pending, amtToTake) : 0;
    require(fee <= maxFeePercentage * amtToTake / WAD, "TeleportJoin/max-fee
        ↪ -exceed");

    // No need of overflow check here as amtToTake is bounded by teleports[
        ↪ hashGUID].pending
    // which is already a uint248. Also int256 >> uint248. Then both
        ↪ castings are safe.
    debt[teleportGUID.sourceDomain] +=  int256(amtToTake);
    teleports[hashGUID].pending     -= uint248(amtToTake);

    if (debt_ >= 0 || uint256(-debt_) < amtToTake) {
        uint256 amtToGenerate = debt_ < 0
```

```
                                    ? uint256(int256(amtToTake) + debt_) //
                                        ↪ amtToTake - |debt_|
                                    : amtToTake;
            // amtToGenerate doesn't need overflow check as it is bounded by
                ↪ amtToTake
            vat.slip(ilk, address(this), int256(amtToGenerate));
            vat.frob(ilk, address(this), address(this), address(this), int256(
                ↪ amtToGenerate), int256(amtToGenerate));
            // Query the actual value as someone might have repaid debt without
                ↪ going through settle (if vat.live == 0 prev frob will revert)
            (, art) = vat.urns(ilk, address(this));
        }
        totalFee = fee + operatorFee;
        postFeeAmount = amtToTake - totalFee;
        daiJoin.exit(bytes32ToAddress(teleportGUID.receiver), postFeeAmount);

        if (fee > 0) {
            vat.move(address(this), vow, fee * RAY);
        }
        if (operatorFee > 0) {
            daiJoin.exit(bytes32ToAddress(teleportGUID.operator), operatorFee);
        }

        emit Mint(hashGUID, teleportGUID, amtToTake, maxFeePercentage,
            ↪ operatorFee, msg.sender);
    }
```

17.2.12 requestMint(teleportGUID, maxFeePercentage, operatorFee) X a

```
    /**
    * @dev External authed function that registers the teleport and executes the
        ↪ mint after
    * @param teleportGUID Struct which contains the whole teleport data
    * @param maxFeePercentage Max percentage of the withdrawn amount (in WAD) to
        ↪ be paid as fee (e.g 1% = 0.01 * WAD)
    * @param operatorFee The amount of DAI to pay to the operator
    * @return postFeeAmount The amount of DAI sent to the receiver after taking
        ↪ out fees
    * @return totalFee The total amount of DAI charged as fees
    **/
    function requestMint(
        TeleportGUID calldata teleportGUID,
        uint256 maxFeePercentage,
        uint256 operatorFee
    ) external auth returns (uint256 postFeeAmount, uint256 totalFee) {
        bytes32 hashGUID = getGUIDHash(teleportGUID);
        require(!teleports[hashGUID].blessed, "TeleportJoin/already-blessed");
        teleports[hashGUID].blessed = true;
        teleports[hashGUID].pending = teleportGUID.amount;
        emit Register(hashGUID, teleportGUID);
        (postFeeAmount, totalFee) = _mint(teleportGUID, hashGUID,
            ↪ maxFeePercentage, operatorFee);
    }
```

17.2.13 mintPending(teleportGUID, maxFeePercentage, operatorFee) X

```
    /**
    * @dev External function that executes the mint of any pending and available
        ↪ amount (only callable by operator or receiver)
    * @param teleportGUID Struct which contains the whole teleport data
    * @param maxFeePercentage Max percentage of the withdrawn amount (in WAD) to
        ↪ be paid as fee (e.g 1% = 0.01 * WAD)
    * @param operatorFee The amount of DAI to pay to the operator
    * @return postFeeAmount The amount of DAI sent to the receiver after taking
        ↪ out fees
    * @return totalFee The total amount of DAI charged as fees
    **/
    function mintPending(
        TeleportGUID calldata teleportGUID,
        uint256 maxFeePercentage,
        uint256 operatorFee
    ) external returns (uint256 postFeeAmount, uint256 totalFee) {
        require(bytes32ToAddress(teleportGUID.receiver) == msg.sender ||
            bytes32ToAddress(teleportGUID.operator) == msg.sender, "TeleportJoin
                ↪ /not-receiver-nor-operator");
        (postFeeAmount, totalFee) = _mint(teleportGUID, getGUIDHash(teleportGUID
            ↪ ), maxFeePercentage, operatorFee);
    }
```

17.2.14 settle(sourceDomain, batchedDaiToFlush) X

```
    /**
    * @dev External function that repays debt with DAI previously pushed to this
        ↪  contract (in general coming from the bridges)
    * @param sourceDomain domain where the DAI is coming from
    * @param batchedDaiToFlush Amount of DAI that is being processed for
        ↪ repayment
    **/
    function settle(bytes32 sourceDomain, uint256 batchedDaiToFlush) external {
        require(batchedDaiToFlush <= uint256(type(int256).max), "TeleportJoin/
            ↪ overflow");
        daiJoin.join(address(this), batchedDaiToFlush);
        if (vat.live() == 1) {
            (, uint256 art_) = vat.urns(ilk, address(this)); // rate == RAY =>
                ↪ normalized debt == actual debt
            uint256 amtToPayBack = _min(batchedDaiToFlush, art_);
            vat.frob(ilk, address(this), address(this), address(this), -int256(
                ↪ amtToPayBack), -int256(amtToPayBack));
            vat.slip(ilk, address(this), -int256(amtToPayBack));
            unchecked {
                art = art_ - amtToPayBack; // Always safe operation
            }
        }
        debt[sourceDomain] -= int256(batchedDaiToFlush);
        emit Settle(sourceDomain, batchedDaiToFlush);
    }
```

17.3 contract TeleportLinearFee

```
contract TeleportLinearFee is TeleportFees {
    uint256 immutable public fee;
    uint256 immutable public ttl;

    uint256 constant public WAD = 10 ** 18;
}
```

17.3.1 constructor(_fee, _ttl) X

```
    /**
    * @param _fee Fee percentage in WAD (e.g 1% fee = 0.01 * WAD)
    * @param _ttl Time in seconds to finalize flush (not teleport)
    **/
    constructor(uint256 _fee, uint256 _ttl) {
        fee = _fee;
        ttl = _ttl;
    }
```

17.3.2 getFee(guid, , , , amtToTake)

```
    function getFee(TeleportGUID calldata guid, uint256, int256, uint256,
       ↪ uint256 amtToTake) override external view returns (uint256) {
        // is slow withdrawal?
        if (block.timestamp >= uint256(guid.timestamp) + ttl) {
            return 0;
        }

        return fee * amtToTake / WAD;
    }
```

17.4 contract TeleportOracleAuth

```
// TeleportOracleAuth provides user authentication for TeleportJoin, by means of
    ↪ Maker Oracle Attestations
contract TeleportOracleAuth {

    mapping (address => uint256) public wards;   // Auth
    mapping (address => uint256) public signers; // Oracle feeds

    TeleportJoinLike immutable public teleportJoin;

    uint256 public threshold;

    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event File(bytes32 indexed what, uint256 data);
    event SignersAdded(address[] signers);
    event SignersRemoved(address[] signers);
}
```

17.4.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "TeleportOracleAuth/not-authorized");
        _;
    }
```

17.4.2 constructor(teleportJoin_) X

```
    constructor(address teleportJoin_) {
        wards[msg.sender] = 1;
        emit Rely(msg.sender);
        teleportJoin = TeleportJoinLike(teleportJoin_);
    }
```

17.4.3 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

17.4.4 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

17.4.5 file(what, data) X a

```
    function file(bytes32 what, uint256 data) external auth {
        if (what == "threshold") {
            threshold = data;
        } else {
            revert("TeleportOracleAuth/file-unrecognized-param");
        }
        emit File(what, data);
    }
```

17.4.6 addSigners(signers_) X a

```
    function addSigners(address[] calldata signers_) external auth {
        for(uint256 i; i < signers_.length; i++) {
            signers[signers_[i]] = 1;
        }
        emit SignersAdded(signers_);
    }
```

17.4.7 removeSigners(signers_) X a

```
    function removeSigners(address[] calldata signers_) external auth {
        for(uint256 i; i < signers_.length; i++) {
            signers[signers_[i]] = 0;
        }
        emit SignersRemoved(signers_);
    }
```

17.4.8 requestMint(teleportGUID, signatures, maxFeePercentage, operatorFee) X

```
    /**
     * @notice Verify oracle signatures and call TeleportJoin to mint DAI if the
        ↪  signatures are valid
     * (only callable by teleport's operator or receiver)
     * @param teleportGUID The teleport GUID to register
     * @param signatures The byte array of concatenated signatures ordered by
        ↪ increasing signer addresses.
     * Each signature is {bytes32 r}{bytes32 s}{uint8 v}
     * @param maxFeePercentage Max percentage of the withdrawn amount (in WAD)
        ↪ to be paid as fee (e.g 1% = 0.01 * WAD)
     * @param operatorFee The amount of DAI to pay to the operator
     * @return postFeeAmount The amount of DAI sent to the receiver after taking
        ↪  out fees
     * @return totalFee The total amount of DAI charged as fees
     */
    function requestMint(
        TeleportGUID calldata teleportGUID,
        bytes calldata signatures,
        uint256 maxFeePercentage,
        uint256 operatorFee
    ) external returns (uint256 postFeeAmount, uint256 totalFee) {
        require(bytes32ToAddress(teleportGUID.receiver) == msg.sender ||
            bytes32ToAddress(teleportGUID.operator) == msg.sender, "
               ↪ TeleportOracleAuth/not-receiver-nor-operator");
        require(isValid(getSignHash(teleportGUID), signatures, threshold), "
           ↪ TeleportOracleAuth/not-enough-valid-sig");
        (postFeeAmount, totalFee) = teleportJoin.requestMint(teleportGUID,
           ↪ maxFeePercentage, operatorFee);
    }
```

17.4.9 isValid(signHash, signatures, threshold_)

```
    /**
     * @notice Returns true if `signatures` contains at least `threshold_` valid
        ↪  signatures of a given `signHash`
     * @param signHash The signed message hash
     * @param signatures The byte array of concatenated signatures ordered by
        ↪ increasing signer addresses.
     * Each signature is {bytes32 r}{bytes32 s}{uint8 v}
     * @param threshold_ The minimum number of valid signatures required for the
        ↪  method to return true
     * @return valid Signature verification result
     */
    function isValid(bytes32 signHash, bytes calldata signatures, uint
        ↪ threshold_) public view returns (bool valid) {
        uint256 count = signatures.length / 65;
        require(count >= threshold_, "TeleportOracleAuth/not-enough-sig");

        uint8 v;
        bytes32 r;
        bytes32 s;
        uint256 numValid;
        address lastSigner;
        for (uint256 i; i < count;) {
            (v,r,s) = splitSignature(signatures, i);
            address recovered = ecrecover(signHash, v, r, s);
            require(recovered > lastSigner, "TeleportOracleAuth/bad-sig-order");
                ↪ // make sure signers are different
            lastSigner = recovered;
            if (signers[recovered] == 1) {
                unchecked { numValid += 1; }
                if (numValid >= threshold_) {
                    return true;
                }
            }
            unchecked { i++; }
        }
    }
```

17.4.10 getSignHash(teleportGUID)

```
    /**
     * @notice This has to match what oracles are signing
     * @param teleportGUID The teleport GUID to calculate hash
     */
    function getSignHash(TeleportGUID memory teleportGUID) public pure returns (
        ↪ bytes32 signHash) {
        signHash = keccak256(abi.encodePacked(
            "\x19Ethereum Signed Message:\n32",
            getGUIDHash(teleportGUID)
        ));
    }
```

17.4.11 splitSignature(signatures, index)

```
    /**
     * @notice Parses the signatures and extract (r, s, v) for a signature at a
         ↪ given index.
     * @param signatures concatenated signatures. Each signature is {bytes32 r}{
         ↪ bytes32 s}{uint8 v}
     * @param index which signature to read (0, 1, 2, ...)
     */
    function splitSignature(bytes calldata signatures, uint256 index) internal
        ↪ pure returns (uint8 v, bytes32 r, bytes32 s) {
        // we jump signatures.offset to get the first slot of signatures content
        // we jump 65 (0x41) per signature
        // for v we load 32 bytes ending with v (the first 31 come from s) then
            ↪ apply a mask
        uint256 start;
        // solhint-disable-next-line no-inline-assembly
        assembly {
            start := mul(0x41, index)
            r := calldataload(add(signatures.offset, start))
            s := calldataload(add(signatures.offset, add(0x20, start)))
            v := and(calldataload(add(signatures.offset, add(0x21, start))), 0
                ↪ xff)
        }
        require(v == 27 || v == 28, "TeleportOracleAuth/bad-v");
    }
```

17.5 contract TeleportRouter

```
contract TeleportRouter {

    using EnumerableSet for EnumerableSet.Bytes32Set;

    mapping (address => uint256) public wards;           // Auth
    mapping (bytes32 => address) public gateways;        // GatewayLike contracts
       ↪  called by the router for each domain
    mapping (address => bytes32) public domains;         // Domains for each
       ↪ gateway

    EnumerableSet.Bytes32Set private allDomains;

    TokenLike immutable public dai; // L1 DAI ERC20 token

    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event File(bytes32 indexed what, bytes32 indexed domain, address data);
}
```

17.5.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "TeleportRouter/not-authorized");
        _;
    }
```

17.5.2 constructor(dai_) X

```
    constructor(address dai_) {
        dai = TokenLike(dai_);
        wards[msg.sender] = 1;
        emit Rely(msg.sender);
    }
```

17.5.3 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

17.5.4 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

17.5.5 file(what, domain, data) X a

```
    /**
     * @notice Allows auth to configure the router. The only supported operation
        ↪  is "gateway",
     * which allows adding, replacing or removing a gateway contract for a given
        ↪  domain. The router forwards `settle()`
     * and `requestMint()` calls to the gateway contract installed for a given
        ↪ domain. Gateway contracts must therefore
     * conform to the GatewayLike interface. Examples of valid gateways include
        ↪ TeleportJoin (for the L1 domain)
     * and L1 bridge contracts (for L2 domains).
     * @dev In addition to updating the mapping ‘gateways‘ which maps
        ↪ GatewayLike contracts to domain names and
     * the reverse mapping ‘domains‘ which maps domain names to GatewayLike
        ↪ contracts, this method also maintains
     * the enumerable set ‘allDomains‘.
     * @param what The name of the operation. Only "gateway" is supported.
     * @param domain The domain for which a GatewayLike contract is added,
        ↪ replaced or removed.
     * @param data The address of the GatewayLike contract to install for the
        ↪ domain (or address(0) to remove a domain)
     */
    function file(bytes32 what, bytes32 domain, address data) external auth {
        if (what == "gateway") {
            address prevGateway = gateways[domain];
            if(prevGateway == address(0)) {
                // new domain => add it to allDomains
                if(data != address(0)) {
                    allDomains.add(domain);
                }
            } else {
                // existing domain
                domains[prevGateway] = bytes32(0);
                if(data == address(0)) {
                    // => remove domain from allDomains
                    allDomains.remove(domain);
                }
            }

            gateways[domain] = data;
            if(data != address(0)) {
                domains[data] = domain;
            }
        } else {
            revert("TeleportRouter/file-unrecognized-param");
        }
        emit File(what, domain, data);
    }
```

17.5.6 numDomains()

```
    function numDomains() external view returns (uint256) {
        return allDomains.length();
    }
```

17.5.7 domainAt(index)

```
    function domainAt(uint256 index) external view returns (bytes32) {
        return allDomains.at(index);
    }
```

17.5.8 hasDomain(domain)

```
    function hasDomain(bytes32 domain) external view returns (bool) {
        return allDomains.contains(domain);
    }
```

17.5.9 requestMint(teleportGUID, maxFeePercentage, operatorFee) X

```
/**
 * @notice Call a GatewayLike contract to request the minting of DAI. The
    ↪ sender must be a supported gateway
 * @param teleportGUID The teleport GUID to register
 * @param maxFeePercentage Max percentage of the withdrawn amount (in WAD)
    ↪ to be paid as fee (e.g 1% = 0.01 * WAD)
 * @param operatorFee The amount of DAI to pay to the operator
 * @return postFeeAmount The amount of DAI sent to the receiver after taking
    ↪  out fees
 * @return totalFee The total amount of DAI charged as fees
 */
function requestMint(
    TeleportGUID calldata teleportGUID,
    uint256 maxFeePercentage,
    uint256 operatorFee
) external returns (uint256 postFeeAmount, uint256 totalFee) {
    require(msg.sender == gateways[teleportGUID.sourceDomain], "
        ↪ TeleportRouter/sender-not-gateway");
    address gateway = gateways[teleportGUID.targetDomain];
    require(gateway != address(0), "TeleportRouter/unsupported-target-domain
        ↪ ");
    (postFeeAmount, totalFee) = GatewayLike(gateway).requestMint(
        ↪ teleportGUID, maxFeePercentage, operatorFee);
}
```

17.5.10 settle(targetDomain, batchedDaiToFlush) X

```
/**
 * @notice Call a GatewayLike contract to settle a batch of sourceDomain ->
    ↪ targetDomain DAI transfer.
 * The sender must be a supported gateway
 * @param targetDomain The domain receiving the batch of DAI (only L1
    ↪ supported for now)
 * @param batchedDaiToFlush The amount of DAI in the batch
 */
function settle(bytes32 targetDomain, uint256 batchedDaiToFlush) external {
    bytes32 sourceDomain = domains[msg.sender];
    require(sourceDomain != bytes32(0), "TeleportRouter/sender-not-gateway")
        ↪ ;
    address gateway = gateways[targetDomain];
    require(gateway != address(0), "TeleportRouter/unsupported-target-domain
        ↪ ");
     // Forward the DAI to settle to the gateway contract
    dai.transferFrom(msg.sender, gateway, batchedDaiToFlush);
    GatewayLike(gateway).settle(sourceDomain, batchedDaiToFlush);
}
```

17.6 contract BasicRelay

```
// Relay messages automatically on the target domain
// User provides gasFee which is paid to the msg.sender
contract BasicRelay {

    mapping (address => uint256) public wards;     // Auth
    mapping (address => uint256) public relayers; // Whitelisted relayers

    DaiJoinLike              public immutable daiJoin;
    TokenLike                public immutable dai;
    TeleportOracleAuthLike public immutable oracleAuth;
    TeleportJoinLike         public immutable teleportJoin;

    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event RelayersAdded(address[] relayers);
    event RelayersRemoved(address[] relayers);

}
```

17.6.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "BasicRelay/not-authorized");
        _;
    }
```

17.6.2 constructor(_oracleAuth, _daiJoin) X

```
    constructor(address _oracleAuth, address _daiJoin) {
        wards[msg.sender] = 1;
        emit Rely(msg.sender);
        oracleAuth = TeleportOracleAuthLike(_oracleAuth);
        daiJoin = DaiJoinLike(_daiJoin);
        dai = daiJoin.dai();
        teleportJoin = oracleAuth.teleportJoin();
    }
```

17.6.3 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

17.6.4 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

17.6.5 addRelayers(relayers_) X a

```
    function addRelayers(address[] calldata relayers_) external auth {
        for(uint256 i; i < relayers_.length; i++) {
            relayers[relayers_[i]] = 1;
        }
        emit RelayersAdded(relayers_);
    }
```

17.6.6 removeRelayers(relayers_) X a

```
    function removeRelayers(address[] calldata relayers_) external auth {
        for(uint256 i; i < relayers_.length; i++) {
            relayers[relayers_[i]] = 0;
        }
        emit RelayersRemoved(relayers_);
    }
```

17.6.7 relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s) X

```
    /**
     * @notice Gasless relay for the Oracle fast path
     * The final signature is ABI-encoded `hashGUID`, `maxFeePercentage`, `
        ↪ gasFee`, `expiry`.
     * Must be called by a whitelisted relayer account with the feeCollector
        ↪ address appended
     * at the end of the calldata, e.g.: `(bool success,) = address(basicRelay).
        ↪ call(abi.encodePacked(relayData, feeCollector));`
     * @param teleportGUID The teleport GUID
     * @param signatures The byte array of concatenated signatures ordered by
        ↪ increasing signer addresses.
     * Each signature is {bytes32 r}{bytes32 s}{uint8 v}
     * @param maxFeePercentage Max percentage of the withdrawn amount (in WAD)
        ↪ to be paid as fee (e.g 1% = 0.01 * WAD)
     * @param gasFee DAI gas fee (in WAD)
     * @param expiry Maximum time for when the query is valid
     * @param v Part of ECDSA signature
     * @param r Part of ECDSA signature
     * @param s Part of ECDSA signature
     */
    function relay(
        TeleportGUID calldata teleportGUID,
        bytes calldata signatures,
        uint256 maxFeePercentage,
        uint256 gasFee,
        uint256 expiry,
        uint8 v,
        bytes32 r,
        bytes32 s
    ) external {
        require(relayers[msg.sender] == 1, "BasicRelay/not-whitelisted");
        require(block.timestamp <= expiry, "BasicRelay/expired");
        bytes32 signHash = keccak256(abi.encodePacked(
            "\x19Ethereum Signed Message:\n32",
            keccak256(abi.encode(getGUIDHash(teleportGUID), maxFeePercentage,
                ↪ gasFee, expiry))
        ));
        address recovered = ecrecover(signHash, v, r, s);
        require(bytes32ToAddress(teleportGUID.receiver) == recovered, "
            ↪ BasicRelay/invalid-signature");

        // Initiate mint and mark the teleport as done
        (uint256 postFeeAmount, uint256 totalFee) = oracleAuth.requestMint(
            ↪ teleportGUID, signatures, maxFeePercentage, gasFee);
        require(postFeeAmount + totalFee == teleportGUID.amount, "BasicRelay/
            ↪ partial-mint-disallowed");

        // Send the gas fee to the fee collector
        address feeCollector;
        // solhint-disable-next-line no-inline-assembly
        assembly {
            feeCollector := shr(96, calldataload(sub(calldatasize(), 20))) //
                ↪ Gelato passes the feeCollector in the same way as in EIP-2771
        }
        dai.transfer(feeCollector, gasFee);
    }
```

17.7 contract TrustedRelay

```
// Relay messages automatically on the target domain
// User provides gasFee which is paid to the msg.sender
// Relay requests are signed by a trusted third-party (typically a backend
    ↪ orchestrating the withdrawal on behalf of the user)
contract TrustedRelay {

    mapping (address => uint256) public wards;    // Auth (Maker governance)
    mapping (address => uint256) public buds;     // Admin accounts managing
        ↪ trusted signers
    mapping (address => uint256) public signers;  // Trusted signers
    mapping (address => uint256) public relayers; // Whitelisted relayers

    uint256                 public gasMargin; // in BPS (e.g 150% = 15000)

    DaiJoinLike             public immutable daiJoin;
    TokenLike               public immutable dai;
    TeleportOracleAuthLike public immutable oracleAuth;
    TeleportJoinLike        public immutable teleportJoin;
    DsValueLike             public immutable ethPriceOracle;

    uint256 constant public WAD_BPS = 10 ** 22; // WAD * BPS = 10^18 * 10^4

    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event Kissed(address indexed usr);
    event Dissed(address indexed usr);
    event File(bytes32 indexed what, uint256 data);
    event SignersAdded(address[] signers);
    event SignersRemoved(address[] signers);
    event RelayersAdded(address[] relayers);
    event RelayersRemoved(address[] relayers);

}
```

17.7.1 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "TrustedRelay/not-authorized");
        _;
    }
```

17.7.2 modifier toll()

```
    modifier toll {
        require(buds[msg.sender] == 1, "TrustedRelay/non-manager");
        _;
    }
```

17.7.3 constructor(_oracleAuth, _daiJoin, _ethPriceOracle) X

```
    constructor(address _oracleAuth, address _daiJoin, address _ethPriceOracle)
        ↪ {
        wards[msg.sender] = 1;
        emit Rely(msg.sender);
        oracleAuth = TeleportOracleAuthLike(_oracleAuth);
        daiJoin = DaiJoinLike(_daiJoin);
        dai = daiJoin.dai();
        teleportJoin = oracleAuth.teleportJoin();
        ethPriceOracle = DsValueLike(_ethPriceOracle);
    }
```

17.7.4 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

17.7.5 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

17.7.6 kiss(usr) X a

```
    function kiss(address usr) external auth {
        buds[usr] = 1;
        emit Kissed(usr);
    }
```

17.7.7 diss(usr) X a

```
    function diss(address usr) external auth {
        buds[usr] = 0;
        emit Dissed(usr);
    }
```

17.7.8 file(what, data) X a

```
    function file(bytes32 what, uint256 data) external auth {
        if (what == "margin") {
            gasMargin = data;
        } else {
            revert("TrustedRelay/file-unrecognized-param");
        }
        emit File(what, data);
    }
```

17.7.9 addRelayers(relayers_) X a

```
    function addRelayers(address[] calldata relayers_) external auth {
        for(uint256 i; i < relayers_.length; i++) {
            relayers[relayers_[i]] = 1;
        }
        emit RelayersAdded(relayers_);
    }
```

17.7.10 removeRelayers(relayers_) X a

```
    function removeRelayers(address[] calldata relayers_) external auth {
        for(uint256 i; i < relayers_.length; i++) {
            relayers[relayers_[i]] = 0;
        }
        emit RelayersRemoved(relayers_);
    }
```

17.7.11 addSigners(signers_) X

```
    function addSigners(address[] calldata signers_) external toll {
        for(uint256 i; i < signers_.length; i++) {
            signers[signers_[i]] = 1;
        }
        emit SignersAdded(signers_);
    }
```

17.7.12 removeSigners(signers_) X

```
    function removeSigners(address[] calldata signers_) external toll {
        for(uint256 i; i < signers_.length; i++) {
            signers[signers_[i]] = 0;
        }
        emit SignersRemoved(signers_);
    }
```

17.7.13 relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s) X

```
    /**
     * @notice Gasless relay for the Oracle fast path
     * The final signature is ABI-encoded `hashGUID`, `maxFeePercentage`, `
        ↪ gasFee`, `expiry`
     * Must be called by a whitelisted relayer account with the feeCollector
        ↪ address appended
     * at the end of the calldata, e.g.: `(bool success,) = address(trustedRelay
        ↪ ).call(abi.encodePacked(relayData, feeCollector));`
     * @param teleportGUID The teleport GUID
     * @param signatures The byte array of concatenated signatures ordered by
        ↪ increasing signer addresses.
     * Each signature is {bytes32 r}{bytes32 s}{uint8 v}
     * @param maxFeePercentage Max percentage of the withdrawn amount (in WAD)
        ↪ to be paid as fee (e.g 1% = 0.01 * WAD)
     * @param gasFee DAI gas fee (in WAD)
     * @param expiry Maximum time for when the query is valid
     * @param v Part of ECDSA signature
     * @param r Part of ECDSA signature
     * @param s Part of ECDSA signature
     */
    function relay(
        TeleportGUID calldata teleportGUID,
        bytes calldata signatures,
        uint256 maxFeePercentage,
        uint256 gasFee,
        uint256 expiry,
        uint8 v,
        bytes32 r,
        bytes32 s
    ) external {
        uint256 startGas = gasleft();

        // Withdraw the L1 DAI to the receiver
        requestMint(teleportGUID, signatures, maxFeePercentage, gasFee, expiry,
            ↪ v, r, s);

        // Send the gas fee to the fee collector
        address feeCollector;
        // solhint-disable-next-line no-inline-assembly
        assembly {
            feeCollector := shr(96, calldataload(sub(calldatasize(), 20))) //
                ↪ Gelato passes the feeCollector in the same way as in EIP-2771
        }
        dai.transfer(feeCollector, gasFee);
        // If the eth price oracle is enabled, use its value to check that
           ↪ gasFee is within an allowable margin
        (bytes32 ethPrice, bool ok) = ethPriceOracle.peek();
        require(!ok || gasFee * WAD_BPS <= uint256(ethPrice) * gasMargin *
           ↪ gasprice() * (startGas - gasleft()), "TrustedRelay/excessive-gas-
           ↪ fee");
    }
```

17.7.14 requestMint(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s)

```
    function requestMint(
        TeleportGUID calldata teleportGUID,
        bytes calldata signatures,
        uint256 maxFeePercentage,
        uint256 gasFee,
        uint256 expiry,
        uint8 v,
        bytes32 r,
        bytes32 s
    ) internal {
        require(relayers[msg.sender] == 1, "TrustedRelay/not-whitelisted");
        require(block.timestamp <= expiry, "TrustedRelay/expired");
        bytes32 signHash = keccak256(abi.encodePacked(
            "\x19Ethereum Signed Message:\n32",
            keccak256(abi.encode(getGUIDHash(teleportGUID), maxFeePercentage,
               ↪ gasFee, expiry))
        ));
        address recovered = ecrecover(signHash, v, r, s);
        require(signers[recovered] == 1 || bytes32ToAddress(teleportGUID.
           ↪ receiver) == recovered, "TrustedRelay/invalid-signature");

        // Initiate mint and mark the teleport as done
        (uint256 postFeeAmount, uint256 totalFee) = oracleAuth.requestMint(
           ↪ teleportGUID, signatures, maxFeePercentage, gasFee);
        require(postFeeAmount + totalFee == teleportGUID.amount, "TrustedRelay/
           ↪ partial-mint-disallowed");
    }
```

17.7.15 gasprice()

```
    function gasprice() internal virtual view returns (uint256) {
        return tx.gasprice;
    }
```

Chapter 18

Ilk registry

18.1 contract GemInfo

```
contract GemInfo {
}
```

18.1.1 name(token)

```
    function name(address token) external view returns (string memory) {
        return TokenLike(token).name();
    }
```

18.1.2 symbol(token)

```
    function symbol(address token) external view returns (string memory) {
        return TokenLike(token).symbol();
    }
```

18.2 contract IlkRegistry

```
contract IlkRegistry {

    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event File(bytes32 indexed what, address data);
    event File(bytes32 indexed ilk, bytes32 indexed what, address data);
    event File(bytes32 indexed ilk, bytes32 indexed what, uint256 data);
    event File(bytes32 indexed ilk, bytes32 indexed what, string data);
    event AddIlk(bytes32 indexed ilk);
    event RemoveIlk(bytes32 indexed ilk);
    event UpdateIlk(bytes32 indexed ilk);
    event NameError(bytes32 indexed ilk);
    event SymbolError(bytes32 indexed ilk);

    // --- Auth ---
    mapping (address => uint) public wards;

    VatLike  public  immutable vat;
    GemInfo  private immutable gemInfo;

    DogLike  public dog;
    CatLike  public cat;
    SpotLike public spot;

    mapping (bytes32 => Ilk) public ilkData;
    bytes32[] ilks;
}
```

18.2.1 struct IlkRegistry.Ilk

```
    struct Ilk {
        uint96  pos;      // Index in ilks array
        address join;     // DSS GemJoin adapter
        address gem;      // The token contract
        uint8   dec;      // Token decimals
        uint96  class;    // Classification code (1 - clip, 2 - flip, 3+ - other)
        address pip;      // Token price
        address xlip;     // Auction contract
        string  name;     // Token name
        string  symbol;   // Token symbol
    }
```

18.2.2 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "IlkRegistry/not-authorized");
        _;
    }
```

18.2.3 rely(usr) X a

```
    function rely(address usr) external auth { wards[usr] = 1; emit Rely(usr); }
```

18.2.4 deny(usr) X a

```
    function deny(address usr) external auth { wards[usr] = 0; emit Deny(usr); }
```

18.2.5 constructor(vat_, dog_, cat_, spot_) X

```
    // Initialize the registry
    constructor(address vat_, address dog_, address cat_, address spot_) public
        ↪ {

        VatLike _vat = vat = VatLike(vat_);
        dog = DogLike(dog_);
        cat = CatLike(cat_);
        spot = SpotLike(spot_);

        require(dog.vat() == vat_,      "IlkRegistry/invalid-dog-vat");
        require(cat.vat() == vat_,      "IlkRegistry/invalid-cat-vat");
        require(spot.vat() == vat_,     "IlkRegistry/invalid-spotter-vat");
        require(_vat.wards(cat_) == 1,  "IlkRegistry/cat-not-authorized");
        require(_vat.wards(spot_) == 1, "IlkRegistry/spot-not-authorized");
        require(_vat.live() == 1,       "IlkRegistry/vat-not-live");
        require(cat.live() == 1,        "IlkRegistry/cat-not-live");
        require(spot.live() == 1,       "IlkRegistry/spot-not-live");

        gemInfo = new GemInfo();

        wards[msg.sender] = 1;
    }
```

18.2.6 add(adapter) X

```
    // Pass an active join adapter to the registry to add it to the set
    function add(address adapter) external {
        JoinLike _join = JoinLike(adapter);

        // Validate adapter
        require(_join.vat() == address(vat),    "IlkRegistry/invalid-join-
            ↪ adapter-vat");
        require(vat.wards(address(_join)) == 1, "IlkRegistry/adapter-not-
            ↪ authorized");

        // Validate ilk
        bytes32 _ilk = _join.ilk();
        require(_ilk != 0, "IlkRegistry/ilk-adapter-invalid");
        require(ilkData[_ilk].join == address(0), "IlkRegistry/ilk-already-
            ↪ exists");

        (address _pip,) = spot.ilks(_ilk);
        require(_pip != address(0), "IlkRegistry/pip-invalid");

        (address _xlip,,,) = dog.ilks(_ilk);

        uint96  _class = 1;
        if (_xlip == address(0)) {
            (_xlip,,)  = cat.ilks(_ilk);
            require(_xlip != address(0), "IlkRegistry/invalid-auction-contract")
                ↪ ;
            _class = 2;
        }

        string memory name = bytes32ToStr(_ilk);
        try gemInfo.name(_join.gem()) returns (string memory _name) {
            if (bytes(_name).length != 0) {
                name = _name;
            }
        } catch {
            emit NameError(_ilk);
        }

        string memory symbol = bytes32ToStr(_ilk);
        try gemInfo.symbol(_join.gem()) returns (string memory _symbol) {
            if (bytes(_symbol).length != 0) {
                symbol = _symbol;
            }
        } catch {
            emit SymbolError(_ilk);
        }

        require(ilks.length < uint96(-1), "IlkRegistry/too-many-ilks");
        ilks.push(_ilk);
        ilkData[ilks[ilks.length - 1]] = Ilk({
            pos: uint96(ilks.length - 1),
            join: address(_join),
            gem: _join.gem(),
            dec: uint8(_join.dec()),
            class: _class,
            pip: _pip,
            xlip: _xlip,
            name: name,
            symbol: symbol
        });

        emit AddIlk(_ilk);
    }
```

18.2.7 remove(ilk) X

```
    // Anyone can remove an ilk if the adapter has been caged
    function remove(bytes32 ilk) external {
        JoinLike _join = JoinLike(ilkData[ilk].join);
        require(address(_join) != address(0), "IlkRegistry/invalid-ilk");
        uint96 _class = ilkData[ilk].class;
        require(_class == 1 || _class == 2, "IlkRegistry/invalid-class");
        require(_join.live() == 0, "IlkRegistry/ilk-live");
        _remove(ilk);
        emit RemoveIlk(ilk);
    }
```

18.2.8 removeAuth(ilk) X a

```
    // Admin can remove an ilk without any precheck
    function removeAuth(bytes32 ilk) external auth {
        _remove(ilk);
        emit RemoveIlk(ilk);
    }
```

18.2.9 file(what, data) X a

```
    // Authed edit function
    function file(bytes32 what, address data) external auth {
        if      (what == "dog")  dog  = DogLike(data);
        else if (what == "cat")  cat  = CatLike(data);
        else if (what == "spot") spot = SpotLike(data);
        else revert("IlkRegistry/file-unrecognized-param-address");
        emit File(what, data);
    }
```

18.2.10 file(ilk, what, data) X a

```
    // Authed edit function
    function file(bytes32 ilk, bytes32 what, address data) external auth {
        if      (what == "gem")  ilkData[ilk].gem  = data;
        else if (what == "join") ilkData[ilk].join = data;
        else if (what == "pip")  ilkData[ilk].pip  = data;
        else if (what == "xlip") ilkData[ilk].xlip = data;
        else revert("IlkRegistry/file-unrecognized-param-address");
        emit File(ilk, what, data);
    }
```

18.2.11 file(ilk, what, data) X a

```
    // Authed edit function
    function file(bytes32 ilk, bytes32 what, uint256 data) external auth {
        if      (what == "class") { require(data <= uint96(-1) && data != 0);
           ↪ ilkData[ilk].class = uint96(data); }
        else if (what == "dec")   { require(data <= uint8(-1));  ilkData[ilk].
           ↪ dec   = uint8(data); }
        else revert("IlkRegistry/file-unrecognized-param-uint256");
        emit File(ilk, what, data);
    }
```

18.2.12 file(ilk, what, data) X a

```
    // Authed edit function
    function file(bytes32 ilk, bytes32 what, string calldata data) external auth
       ↪  {
        if      (what == "name")   ilkData[ilk].name   = data;
        else if (what == "symbol") ilkData[ilk].symbol = data;
        else revert("IlkRegistry/file-unrecognized-param-string");
        emit File(ilk, what, data);
    }
```

18.2.13 _remove(ilk)

```
    // Remove ilk from the ilks array by replacing the ilk with the
    //  last in the array and then trimming the end.
    function _remove(bytes32 ilk) internal {
        // Get the position in the array
        uint256 _index = ilkData[ilk].pos;
        // Get the last ilk in the array
        bytes32 _moveIlk = ilks[ilks.length - 1];
        // Replace the ilk we are removing
        ilks[_index] = _moveIlk;
        // Update the array position for the moved ilk
        ilkData[_moveIlk].pos = uint96(_index);
        // Trim off the end of the ilks array
        ilks.pop();
        // Delete struct data
        delete ilkData[ilk];
    }
```

18.2.14 count()

```
    // The number of active ilks
    function count() external view returns (uint256) {
        return ilks.length;
    }
```

18.2.15 list()

```
    // Return an array of the available ilks
    function list() external view returns (bytes32[] memory) {
        return ilks;
    }
```

18.2.16 list(start, end)

```
// Get a splice of the available ilks, useful when ilks array is large.
function list(uint256 start, uint256 end) external view returns (bytes32[]
    ↪ memory) {
    require(start <= end && end < ilks.length, "IlkRegistry/invalid-input");
    bytes32[] memory _ilks = new bytes32[]((end - start) + 1);
    uint256 _count = 0;
    for (uint256 i = start; i <= end; i++) {
        _ilks[_count] = ilks[i];
        _count++;
    }
    return _ilks;
}
```

18.2.17 get(pos)

```
// Get the ilk at a specific position in the array
function get(uint256 pos) external view returns (bytes32) {
    require(pos < ilks.length, "IlkRegistry/index-out-of-range");
    return ilks[pos];
}
```

18.2.18 info(ilk)

```
// Get information about an ilk, including name and symbol
function info(bytes32 ilk) external view returns (
    string memory name,
    string memory symbol,
    uint256 class,
    uint256 dec,
    address gem,
    address pip,
    address join,
    address xlip
) {
    Ilk memory _ilk = ilkData[ilk];
    return (
        _ilk.name,
        _ilk.symbol,
        _ilk.class,
        _ilk.dec,
        _ilk.gem,
        _ilk.pip,
        _ilk.join,
        _ilk.xlip
    );
}
```

18.2.19 pos(ilk)

```
// The location of the ilk in the ilks array
function pos(bytes32 ilk) external view returns (uint256) {
    return ilkData[ilk].pos;
}
```

18.2.20 class(ilk)

```
// The classification code of the ilk
//   1  - Clipper
//   2  - Flipper
//   3+ - RWA or custom adapter
```

```
    function class(bytes32 ilk) external view returns (uint256) {
        return ilkData[ilk].class;
    }
```

18.2.21 gem(ilk)

```
    // The token address
    function gem(bytes32 ilk) external view returns (address) {
        return ilkData[ilk].gem;
    }
```

18.2.22 pip(ilk)

```
    // The ilk's price feed
    function pip(bytes32 ilk) external view returns (address) {
        return ilkData[ilk].pip;
    }
```

18.2.23 join(ilk)

```
    // The ilk's join adapter
    function join(bytes32 ilk) external view returns (address) {
        return ilkData[ilk].join;
    }
```

18.2.24 xlip(ilk)

```
    // The auction contract for the ilk
    function xlip(bytes32 ilk) external view returns (address) {
        return ilkData[ilk].xlip;
    }
```

18.2.25 dec(ilk)

```
    // The number of decimals on the ilk
    function dec(bytes32 ilk) external view returns (uint256) {
        return ilkData[ilk].dec;
    }
```

18.2.26 symbol(ilk)

```
    // Return the symbol of the token, if available
    function symbol(bytes32 ilk) external view returns (string memory) {
        return ilkData[ilk].symbol;
    }
```

18.2.27 name(ilk)

```
    // Return the name of the token, if available
    function name(bytes32 ilk) external view returns (string memory) {
        return ilkData[ilk].name;
    }
```

18.2.28 update(ilk) X

```
// Public function to update an ilk's pip and flip if the ilk has been
    ↪ updated.
function update(bytes32 ilk) external {
    require(JoinLike(ilkData[ilk].join).vat() == address(vat), "IlkRegistry/
        ↪ invalid-ilk");
    require(JoinLike(ilkData[ilk].join).live() == 1, "IlkRegistry/ilk-not-
        ↪ live-use-remove-instead");
    uint96 _class = ilkData[ilk].class;
    require(_class == 1 || _class == 2, "IlkRegistry/invalid-class");

    (address _pip,) = spot.ilks(ilk);
    require(_pip != address(0), "IlkRegistry/pip-invalid");

    ilkData[ilk].pip    = _pip;
    emit UpdateIlk(ilk);
}
```

18.2.29 put(_ilk, _join, _gem, _dec, _class, _pip, _xlip, _name, _symbol) X a

```
// Force addition or update of a collateral type. (i.e. for RWA, etc.)
//  Governance managed
function put(
        bytes32 _ilk,
        address _join,
        address _gem,
        uint256 _dec,
        uint256 _class,
        address _pip,
        address _xlip,
        string calldata _name,
        string calldata _symbol
        )
    external auth {

        require(_class != 0 && _class <= uint96(-1), "IlkRegistry/invalid-
            ↪ class");
        require(_dec <= uint8(-1), "IlkRegistry/invalid-dec");
        uint96 _pos;

        if (ilkData[_ilk].class == 0) {
            require(ilks.length < uint96(-1), "IlkRegistry/too-many-ilks");
            ilks.push(_ilk);
            _pos = uint96(ilks.length - 1);
            emit AddIlk(_ilk);
        } else {
            _pos = ilkData[_ilk].pos;
            emit UpdateIlk(_ilk);
        }

        ilkData[ilks[_pos]] = Ilk({
            pos: _pos,
            join: _join,
            gem: _gem,
            dec: uint8(_dec),
            class: uint96(_class),
            pip: _pip,
            xlip: _xlip,
            name: _name,
            symbol: _symbol
        });
}
```

18.2.30 bytes32ToStr(_bytes32)

```
function bytes32ToStr(bytes32 _bytes32) internal pure returns (string memory
    ↪ ) {
    bytes memory _bytesArray = new bytes(32);
    for (uint256 i; i < 32; i++) {
        _bytesArray[i] = _bytes32[i];
    }
    return string(_bytesArray);
}
```

Chapter 19

DSS Vest

19.1 contract DssVestMintable

```
contract DssVestMintable is DssVest {

    MintLike public immutable gem;
}
```

Inherited:

```
abstract contract DssVest {
    // --- Data ---
    mapping (address => uint256) public wards;
    mapping (uint256 => Award) public awards;

    uint256 public cap; // Maximum per-second issuance token rate

    uint256 public ids; // Total vestings

    uint256 internal locked;

    uint256 public constant  TWENTY_YEARS = 20 * 365 days;

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);

    event File(bytes32 indexed what, uint256 data);

    event Init(uint256 indexed id, address indexed usr);
    event Vest(uint256 indexed id, uint256 amt);
    event Restrict(uint256 indexed id);
    event Unrestrict(uint256 indexed id);
    event Yank(uint256 indexed id, uint256 end);
    event Move(uint256 indexed id, address indexed dst);

    /**
        @dev Override this to implement payment logic.
        @param _guy The payment target.
        @param _amt The payment amount. [units are implementation-specific]
    */
    function pay(address _guy, uint256 _amt) virtual internal;
}
```

19.1.1 struct DssVest.Award

```
    struct Award {
        address usr;   // Vesting recipient
        uint48  bgn;   // Start of vesting period  [timestamp]
        uint48  clf;   // The cliff date           [timestamp]
        uint48  fin;   // End of vesting period    [timestamp]
        address mgr;   // A manager address that can yank
        uint8   res;   // Restricted
        uint128 tot;   // Total reward amount
        uint128 rxd;   // Amount of vest claimed
    }
```

19.1.2 modifier lock() [DssVest]

```
    // --- Mutex ---
    modifier lock {
        require(locked == 0, "DssVest/system-locked");
        locked = 1;
        _;
        locked = 0;
    }
```

19.1.3 modifier auth() [DssVest]

```
    // --- Auth ---
    modifier auth {
        require(wards[msg.sender] == 1, "DssVest/not-authorized");
        _;
    }
```

19.1.4 usr(_id) [DssVest]

```
    // Getters to access only to the value desired
    function usr(uint256 _id) external view returns (address) {
        return awards[_id].usr;
    }
```

19.1.5 bgn(_id) [DssVest]

```
    function bgn(uint256 _id) external view returns (uint256) {
        return awards[_id].bgn;
    }
```

19.1.6 clf(_id) [DssVest]

```
    function clf(uint256 _id) external view returns (uint256) {
        return awards[_id].clf;
    }
```

19.1.7 fin(_id) [DssVest]

```
    function fin(uint256 _id) external view returns (uint256) {
        return awards[_id].fin;
    }
```

19.1.8 mgr(_id) [DssVest]

```
    function mgr(uint256 _id) external view returns (address) {
        return awards[_id].mgr;
    }
```

19.1.9 res(_id) [DssVest]

```
function res(uint256 _id) external view returns (uint256) {
    return awards[_id].res;
}
```

19.1.10 tot(_id) [DssVest]

```
function tot(uint256 _id) external view returns (uint256) {
    return awards[_id].tot;
}
```

19.1.11 rxd(_id) [DssVest]

```
function rxd(uint256 _id) external view returns (uint256) {
    return awards[_id].rxd;
}
```

19.1.12 constructor() [DssVest] X

```
/**
    @dev Base vesting logic contract constructor
*/
constructor() public {
    wards[msg.sender] = 1;
    emit Rely(msg.sender);
}
```

19.1.13 rely(_usr) [DssVest] X a

```
function rely(address _usr) external auth {
    wards[_usr] = 1;
    emit Rely(_usr);
}
```

19.1.14 deny(_usr) [DssVest] X a

```
function deny(address _usr) external auth {
    wards[_usr] = 0;
    emit Deny(_usr);
}
```

19.1.15 file(what, data) [DssVest] X a

```
/**
    @dev (Required) Set the per-second token issuance rate.
    @param what  The tag of the value to change (ex. bytes32("cap"))
    @param data  The value to update (ex. cap of 1000 tokens/yr == 1000*WAD
      ↪ /365 days)
*/
function file(bytes32 what, uint256 data) external lock auth {
    if      (what == "cap")          cap = data;     // The maximum amount of
      ↪ tokens that can be streamed per-second per vest
    else revert("DssVest/file-unrecognized-param");
    emit File(what, data);
}
```

19.1.16 min(x, y) [DssVest]

```
    function min(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = x > y ? y : x;
    }
```

19.1.17 add(x, y) [DssVest]

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "DssVest/add-overflow");
    }
```

19.1.18 sub(x, y) [DssVest]

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "DssVest/sub-underflow");
    }
```

19.1.19 mul(x, y) [DssVest]

```
    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require(y == 0 || (z = x * y) / y == x, "DssVest/mul-overflow");
    }
```

19.1.20 toUint48(x) [DssVest]

```
    function toUint48(uint256 x) internal pure returns (uint48 z) {
        require((z = uint48(x)) == x, "DssVest/uint48-overflow");
    }
```

19.1.21 toUint128(x) [DssVest]

```
    function toUint128(uint256 x) internal pure returns (uint128 z) {
        require((z = uint128(x)) == x, "DssVest/uint128-overflow");
    }
```

19.1.22 create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a

```
    /**
        @dev Govanance adds a vesting contract
        @param _usr The recipient of the reward
        @param _tot The total amount of the vest
        @param _bgn The starting timestamp of the vest
        @param _tau The duration of the vest (in seconds)
        @param _eta The cliff duration in seconds (i.e. 1 years)
        @param _mgr An optional manager for the contract. Can yank if vesting
           ↪ ends prematurely.
        @return id  The id of the vesting contract
    */
    function create(address _usr, uint256 _tot, uint256 _bgn, uint256 _tau,
       ↪ uint256 _eta, address _mgr) external lock auth returns (uint256 id) {
        require(_usr != address(0),                        "DssVest/invalid-user
           ↪ ");
        require(_tot > 0,                                  "DssVest/no-vest-
           ↪ total-amount");
        require(_bgn < add(block.timestamp, TWENTY_YEARS), "DssVest/bgn-too-far"
           ↪ );
        require(_bgn > sub(block.timestamp, TWENTY_YEARS), "DssVest/bgn-too-long
           ↪ -ago");
        require(_tau > 0,                                        "DssVest/tau-zero");
        require(_tot / _tau <= cap,                              "DssVest/rate-too-
            ↪ high");
        require(_tau <= TWENTY_YEARS,                            "DssVest/tau-too-long
            ↪ ");
        require(_eta <= _tau,                                    "DssVest/eta-too-long
            ↪ ");
        require(ids < type(uint256).max,                         "DssVest/ids-overflow
            ↪ ");

        id = ++ids;
        awards[id] = Award({
            usr: _usr,
            bgn: toUint48(_bgn),
            clf: toUint48(add(_bgn, _eta)),
            fin: toUint48(add(_bgn, _tau)),
            tot: toUint128(_tot),
            rxd: 0,
            mgr: _mgr,
            res: 0
        });
        emit Init(id, _usr);
    }
```

19.1.23 vest(_id) [DssVest] X

```
    /**
        @dev Anyone (or only owner of a vesting contract if restricted) calls
            ↪ this to claim all available rewards
        @param _id     The id of the vesting contract
    */
    function vest(uint256 _id) external {
        _vest(_id, type(uint256).max);
    }
```

19.1.24 vest(_id, _maxAmt) [DssVest] X

```
    /**
        @dev Anyone (or only owner of a vesting contract if restricted) calls
            ↪ this to claim rewards
        @param _id     The id of the vesting contract
        @param _maxAmt The maximum amount to vest
    */
    function vest(uint256 _id, uint256 _maxAmt) external {
        _vest(_id, _maxAmt);
    }
```

19.1.25 _vest(_id, _maxAmt) [DssVest]

```
    /**
        @dev Anyone (or only owner of a vesting contract if restricted) calls
            ↪ this to claim rewards
        @param _id     The id of the vesting contract
        @param _maxAmt The maximum amount to vest
    */
    function _vest(uint256 _id, uint256 _maxAmt) internal lock {
        Award memory _award = awards[_id];
        require(_award.usr != address(0), "DssVest/invalid-award");
        require(_award.res == 0 || _award.usr == msg.sender, "DssVest/only-user-
            ↪ can-claim");
        uint256 amt = unpaid(block.timestamp, _award.bgn, _award.clf, _award.fin
            ↪ , _award.tot, _award.rxd);
        amt = min(amt, _maxAmt);
        awards[_id].rxd = toUint128(add(_award.rxd, amt));
        pay(_award.usr, amt);
        emit Vest(_id, amt);
    }
```

19.1.26 accrued(_id) [DssVest]

```
    /**
        @dev amount of tokens accrued, not accounting for tokens paid
        @param _id  The id of the vesting contract
        @return amt The accrued amount
    */
    function accrued(uint256 _id) external view returns (uint256 amt) {
        Award memory _award = awards[_id];
        require(_award.usr != address(0), "DssVest/invalid-award");
        amt = accrued(block.timestamp, _award.bgn, _award.fin, _award.tot);
    }
```

19.1.27 accrued(_time, _bgn, _fin, _tot) [DssVest]

```
    /**
        @dev amount of tokens accrued, not accounting for tokens paid
        @param _time The timestamp to perform the calculation
        @param _bgn  The start time of the contract
        @param _fin  The end time of the contract
        @param _tot  The total amount of the contract
        @return amt  The accrued amount
    */
    function accrued(uint256 _time, uint48 _bgn, uint48 _fin, uint128 _tot)
       ↪ internal pure returns (uint256 amt) {
        if (_time < _bgn) {
            amt = 0;
        } else if (_time >= _fin) {
            amt = _tot;
        } else {
            amt = mul(_tot, sub(_time, _bgn)) / sub(_fin, _bgn); // 0 <= amt <
                ↪ _award.tot
        }
    }
```

19.1.28 unpaid(_id) [DssVest]

```
    /**
        @dev return the amount of vested, claimable GEM for a given ID
        @param _id  The id of the vesting contract
        @return amt The claimable amount
    */
    function unpaid(uint256 _id) external view returns (uint256 amt) {
        Award memory _award = awards[_id];
        require(_award.usr != address(0), "DssVest/invalid-award");
        amt = unpaid(block.timestamp, _award.bgn, _award.clf, _award.fin, _award
            ↪ .tot, _award.rxd);
    }
```

19.1.29 unpaid(_time, _bgn, _clf, _fin, _tot, _rxd) [DssVest]

```
    /**
        @dev amount of tokens accrued, not accounting for tokens paid
        @param _time The timestamp to perform the calculation
        @param _bgn  The start time of the contract
        @param _clf  The timestamp of the cliff
        @param _fin  The end time of the contract
        @param _tot  The total amount of the contract
        @param _rxd  The number of gems received
        @return amt  The claimable amount
    */
    function unpaid(uint256 _time, uint48 _bgn, uint48 _clf, uint48 _fin,
        ↪ uint128 _tot, uint128 _rxd) internal pure returns (uint256 amt) {
        amt = _time < _clf ? 0 : sub(accrued(_time, _bgn, _fin, _tot), _rxd);
    }
```

19.1.30 restrict(_id) [DssVest] X

```
    /**
        @dev Allows governance or the owner to restrict vesting to the owner
            ↪ only
        @param _id The id of the vesting contract
    */
    function restrict(uint256 _id) external lock {
        address usr_ = awards[_id].usr;
        require(usr_ != address(0), "DssVest/invalid-award");
        require(wards[msg.sender] == 1 || usr_ == msg.sender, "DssVest/not-
            ↪ authorized");
        awards[_id].res = 1;
        emit Restrict(_id);
    }
```

19.1.31 unrestrict(_id) [DssVest] X

```
    /**
        @dev Allows governance or the owner to enable permissionless vesting
        @param _id The id of the vesting contract
    */
    function unrestrict(uint256 _id) external lock {
        address usr_ = awards[_id].usr;
        require(usr_ != address(0), "DssVest/invalid-award");
        require(wards[msg.sender] == 1 || usr_ == msg.sender, "DssVest/not-
            ↪ authorized");
        awards[_id].res = 0;
        emit Unrestrict(_id);
    }
```

19.1.32 yank(_id) [DssVest] X

```
    /**
        @dev Allows governance or the manager to remove a vesting contract
            ↪ immediately
        @param _id The id of the vesting contract
    */
    function yank(uint256 _id) external {
        _yank(_id, block.timestamp);
    }
```

19.1.33 yank(_id, _end) [DssVest] X

```
    /**
        @dev Allows governance or the manager to remove a vesting contract at a
            ↪ future time
        @param _id  The id of the vesting contract
        @param _end A scheduled time to end the vest
    */
    function yank(uint256 _id, uint256 _end) external {
        _yank(_id, _end);
    }
```

19.1.34 _yank(_id, _end) [DssVest]

```
/**
    @dev Allows governance or the manager to end pre-maturely a vesting
        ↪ contract
    @param _id  The id of the vesting contract
    @param _end A scheduled time to end the vest
*/
function _yank(uint256 _id, uint256 _end) internal lock {
    require(wards[msg.sender] == 1 || awards[_id].mgr == msg.sender, "
        ↪ DssVest/not-authorized");
    Award memory _award = awards[_id];
    require(_award.usr != address(0), "DssVest/invalid-award");
    if (_end < block.timestamp) {
        _end = block.timestamp;
    }
    if (_end < _award.fin) {
        uint48 end = toUint48(_end);
        awards[_id].fin = end;
        if (end < _award.bgn) {
            awards[_id].bgn = end;
            awards[_id].clf = end;
            awards[_id].tot = 0;
        } else if (end < _award.clf) {
            awards[_id].clf = end;
            awards[_id].tot = 0;
        } else {
            awards[_id].tot = toUint128(
                                add(
                                    unpaid(_end, _award.bgn, _award.clf,
                                        ↪ _award.fin, _award.tot, _award.
                                        ↪ rxd),
                                    _award.rxd
                                )
                            );
        }
    }

    emit Yank(_id, _end);
}
```

19.1.35 move(_id, _dst) [DssVest] X

```
/**
    @dev Allows owner to move a contract to a different address
    @param _id  The id of the vesting contract
    @param _dst The address to send ownership of the contract to
*/
function move(uint256 _id, address _dst) external lock {
    require(awards[_id].usr == msg.sender, "DssVest/only-user-can-move");
    require(_dst != address(0), "DssVest/zero-address-invalid");
    awards[_id].usr = _dst;
    emit Move(_id, _dst);
}
```

19.1.36 valid(_id) [DssVest]

```
/**
    @dev Return true if a contract is valid
    @param _id The id of the vesting contract
    @return isValid True for valid contract
*/
function valid(uint256 _id) external view returns (bool isValid) {
    isValid = awards[_id].rxd < awards[_id].tot;
}
```

19.1.37 constructor(_gem) X

```
/**
    @dev This contract must be authorized to 'mint' on the token
    @param _gem The contract address of the mintable token
*/
constructor(address _gem) public DssVest() {
    require(_gem != address(0), "DssVestMintable/Invalid-token-address");
    gem = MintLike(_gem);
}
```

19.1.38 pay(_guy, _amt)

```
/**
    @dev Override pay to handle mint logic
    @param _guy The recipient of the minted token
    @param _amt The amount of token units to send to the _guy
*/
function pay(address _guy, uint256 _amt) override internal {
    gem.mint(_guy, _amt);
}
```

19.2 contract DssVestSuckable

```
contract DssVestSuckable is DssVest {

    uint256 internal constant RAY = 10**27;

    ChainlogLike public immutable chainlog;
    VatLike      public immutable vat;
    DaiJoinLike  public immutable daiJoin;
}
```

Inherited:

```
abstract contract DssVest {
    // --- Data ---
    mapping (address => uint256) public wards;
    mapping (uint256 => Award) public awards;

    uint256 public cap; // Maximum per-second issuance token rate

    uint256 public ids; // Total vestings

    uint256 internal locked;

    uint256 public constant  TWENTY_YEARS = 20 * 365 days;

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);

    event File(bytes32 indexed what, uint256 data);

    event Init(uint256 indexed id, address indexed usr);
    event Vest(uint256 indexed id, uint256 amt);
    event Restrict(uint256 indexed id);
    event Unrestrict(uint256 indexed id);
    event Yank(uint256 indexed id, uint256 end);
    event Move(uint256 indexed id, address indexed dst);

    /**
        @dev Override this to implement payment logic.
        @param _guy The payment target.
        @param _amt The payment amount. [units are implementation-specific]
    */
    function pay(address _guy, uint256 _amt) virtual internal;
}
```

19.2.1 struct DssVest.Award

```
    struct Award {
        address usr;   // Vesting recipient
        uint48  bgn;   // Start of vesting period  [timestamp]
        uint48  clf;   // The cliff date           [timestamp]
        uint48  fin;   // End of vesting period    [timestamp]
        address mgr;   // A manager address that can yank
        uint8   res;   // Restricted
        uint128 tot;   // Total reward amount
        uint128 rxd;   // Amount of vest claimed
    }
```

19.2.2 modifier lock() [DssVest]

```
    // --- Mutex ---
    modifier lock {
        require(locked == 0, "DssVest/system-locked");
        locked = 1;
        _;
        locked = 0;
    }
```

19.2.3 modifier auth() [DssVest]

```
    // --- Auth ---
    modifier auth {
        require(wards[msg.sender] == 1, "DssVest/not-authorized");
        _;
    }
```

19.2.4 usr(_id) [DssVest]

```
    // Getters to access only to the value desired
    function usr(uint256 _id) external view returns (address) {
        return awards[_id].usr;
    }
```

19.2.5 bgn(_id) [DssVest]

```
    function bgn(uint256 _id) external view returns (uint256) {
        return awards[_id].bgn;
    }
```

19.2.6 clf(_id) [DssVest]

```
    function clf(uint256 _id) external view returns (uint256) {
        return awards[_id].clf;
    }
```

19.2.7 fin(_id) [DssVest]

```
    function fin(uint256 _id) external view returns (uint256) {
        return awards[_id].fin;
    }
```

19.2.8 mgr(_id) [DssVest]

```
    function mgr(uint256 _id) external view returns (address) {
        return awards[_id].mgr;
    }
```

19.2.9 res(_id) [DssVest]

```
    function res(uint256 _id) external view returns (uint256) {
        return awards[_id].res;
    }
```

19.2.10 tot(_id) [DssVest]

```
    function tot(uint256 _id) external view returns (uint256) {
        return awards[_id].tot;
    }
```

19.2.11 rxd(_id) [DssVest]

```
function rxd(uint256 _id) external view returns (uint256) {
    return awards[_id].rxd;
}
```

19.2.12 constructor() [DssVest] X

```
/**
    @dev Base vesting logic contract constructor
*/
constructor() public {
    wards[msg.sender] = 1;
    emit Rely(msg.sender);
}
```

19.2.13 rely(_usr) [DssVest] X a

```
function rely(address _usr) external auth {
    wards[_usr] = 1;
    emit Rely(_usr);
}
```

19.2.14 deny(_usr) [DssVest] X a

```
function deny(address _usr) external auth {
    wards[_usr] = 0;
    emit Deny(_usr);
}
```

19.2.15 file(what, data) [DssVest] X a

```
/**
    @dev (Required) Set the per-second token issuance rate.
    @param what  The tag of the value to change (ex. bytes32("cap"))
    @param data  The value to update (ex. cap of 1000 tokens/yr == 1000*WAD
       ↪ /365 days)
*/
function file(bytes32 what, uint256 data) external lock auth {
    if      (what == "cap")         cap = data;     // The maximum amount of
       ↪ tokens that can be streamed per-second per vest
    else revert("DssVest/file-unrecognized-param");
    emit File(what, data);
}
```

19.2.16 min(x, y) [DssVest]

```
function min(uint256 x, uint256 y) internal pure returns (uint256 z) {
    z = x > y ? y : x;
}
```

19.2.17 add(x, y) [DssVest]

```
function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
    require((z = x + y) >= x, "DssVest/add-overflow");
}
```

19.2.18 sub(x, y) [DssVest]

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "DssVest/sub-underflow");
    }
```

19.2.19 mul(x, y) [DssVest]

```
    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require(y == 0 || (z = x * y) / y == x, "DssVest/mul-overflow");
    }
```

19.2.20 toUint48(x) [DssVest]

```
    function toUint48(uint256 x) internal pure returns (uint48 z) {
        require((z = uint48(x)) == x, "DssVest/uint48-overflow");
    }
```

19.2.21 toUint128(x) [DssVest]

```
    function toUint128(uint256 x) internal pure returns (uint128 z) {
        require((z = uint128(x)) == x, "DssVest/uint128-overflow");
    }
```

19.2.22 create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a

```
    /**
        @dev Govanance adds a vesting contract
        @param _usr The recipient of the reward
        @param _tot The total amount of the vest
        @param _bgn The starting timestamp of the vest
        @param _tau The duration of the vest (in seconds)
        @param _eta The cliff duration in seconds (i.e. 1 years)
        @param _mgr An optional manager for the contract. Can yank if vesting
            ↪ ends prematurely.
        @return id  The id of the vesting contract
    */
    function create(address _usr, uint256 _tot, uint256 _bgn, uint256 _tau,
        ↪ uint256 _eta, address _mgr) external lock auth returns (uint256 id) {
        require(_usr != address(0),                        "DssVest/invalid-user
            ↪ ");
        require(_tot > 0,                                  "DssVest/no-vest-
            ↪ total-amount");
        require(_bgn < add(block.timestamp, TWENTY_YEARS), "DssVest/bgn-too-far"
            ↪ );
        require(_bgn > sub(block.timestamp, TWENTY_YEARS), "DssVest/bgn-too-long
            ↪ -ago");
        require(_tau > 0,                                  "DssVest/tau-zero");
        require(_tot / _tau <= cap,                        "DssVest/rate-too-
            ↪ high");
        require(_tau <= TWENTY_YEARS,                      "DssVest/tau-too-long
            ↪ ");
        require(_eta <= _tau,                              "DssVest/eta-too-long
            ↪ ");
        require(ids < type(uint256).max,                   "DssVest/ids-overflow
            ↪ ");

        id = ++ids;
        awards[id] = Award({
            usr: _usr,
            bgn: toUint48(_bgn),
            clf: toUint48(add(_bgn, _eta)),
            fin: toUint48(add(_bgn, _tau)),
            tot: toUint128(_tot),
            rxd: 0,
            mgr: _mgr,
            res: 0
        });
        emit Init(id, _usr);
    }
```

19.2.23 vest(_id) [DssVest] X

```
    /**
        @dev Anyone (or only owner of a vesting contract if restricted) calls
            ↪ this to claim all available rewards
        @param _id     The id of the vesting contract
    */
    function vest(uint256 _id) external {
        _vest(_id, type(uint256).max);
    }
```

19.2.24 vest(_id, _maxAmt) [DssVest] X

```
    /**
        @dev Anyone (or only owner of a vesting contract if restricted) calls
            ↪ this to claim rewards
        @param _id     The id of the vesting contract
        @param _maxAmt The maximum amount to vest
    */
    function vest(uint256 _id, uint256 _maxAmt) external {
        _vest(_id, _maxAmt);
    }
```

19.2.25 _vest(_id, _maxAmt) [DssVest]

```
    /**
        @dev Anyone (or only owner of a vesting contract if restricted) calls
            ↪ this to claim rewards
        @param _id     The id of the vesting contract
        @param _maxAmt The maximum amount to vest
    */
    function _vest(uint256 _id, uint256 _maxAmt) internal lock {
        Award memory _award = awards[_id];
        require(_award.usr != address(0), "DssVest/invalid-award");
        require(_award.res == 0 || _award.usr == msg.sender, "DssVest/only-user-
            ↪ can-claim");
        uint256 amt = unpaid(block.timestamp, _award.bgn, _award.clf, _award.fin
            ↪ , _award.tot, _award.rxd);
        amt = min(amt, _maxAmt);
        awards[_id].rxd = toUint128(add(_award.rxd, amt));
        pay(_award.usr, amt);
        emit Vest(_id, amt);
    }
```

19.2.26 accrued(_id) [DssVest]

```
    /**
        @dev amount of tokens accrued, not accounting for tokens paid
        @param _id  The id of the vesting contract
        @return amt The accrued amount
    */
    function accrued(uint256 _id) external view returns (uint256 amt) {
        Award memory _award = awards[_id];
        require(_award.usr != address(0), "DssVest/invalid-award");
        amt = accrued(block.timestamp, _award.bgn, _award.fin, _award.tot);
    }
```

19.2.27 accrued(_time, _bgn, _fin, _tot) [DssVest]

```
    /**
        @dev amount of tokens accrued, not accounting for tokens paid
        @param _time The timestamp to perform the calculation
        @param _bgn  The start time of the contract
        @param _fin  The end time of the contract
        @param _tot  The total amount of the contract
        @return amt  The accrued amount
    */
    function accrued(uint256 _time, uint48 _bgn, uint48 _fin, uint128 _tot)
        ↪ internal pure returns (uint256 amt) {
        if (_time < _bgn) {
            amt = 0;
        } else if (_time >= _fin) {
            amt = _tot;
        } else {
            amt = mul(_tot, sub(_time, _bgn)) / sub(_fin, _bgn); // 0 <= amt <
                ↪ _award.tot
        }
    }
```

19.2.28 unpaid(_id) [DssVest]

```
    /**
        @dev return the amount of vested, claimable GEM for a given ID
        @param _id  The id of the vesting contract
        @return amt The claimable amount
    */
    function unpaid(uint256 _id) external view returns (uint256 amt) {
        Award memory _award = awards[_id];
        require(_award.usr != address(0), "DssVest/invalid-award");
        amt = unpaid(block.timestamp, _award.bgn, _award.clf, _award.fin, _award
            ↪ .tot, _award.rxd);
    }
```

19.2.29 unpaid(_time, _bgn, _clf, _fin, _tot, _rxd) [DssVest]

```
    /**
        @dev amount of tokens accrued, not accounting for tokens paid
        @param _time The timestamp to perform the calculation
        @param _bgn  The start time of the contract
        @param _clf  The timestamp of the cliff
        @param _fin  The end time of the contract
        @param _tot  The total amount of the contract
        @param _rxd  The number of gems received
        @return amt  The claimable amount
    */
    function unpaid(uint256 _time, uint48 _bgn, uint48 _clf, uint48 _fin,
        ↪ uint128 _tot, uint128 _rxd) internal pure returns (uint256 amt) {
        amt = _time < _clf ? 0 : sub(accrued(_time, _bgn, _fin, _tot), _rxd);
    }
```

19.2.30 restrict(_id) [DssVest] X

```
    /**
        @dev Allows governance or the owner to restrict vesting to the owner
            ↪ only
        @param _id The id of the vesting contract
    */
    function restrict(uint256 _id) external lock {
        address usr_ = awards[_id].usr;
        require(usr_ != address(0), "DssVest/invalid-award");
        require(wards[msg.sender] == 1 || usr_ == msg.sender, "DssVest/not-
            ↪ authorized");
        awards[_id].res = 1;
        emit Restrict(_id);
    }
```

19.2.31 unrestrict(_id) [DssVest] X

```
    /**
        @dev Allows governance or the owner to enable permissionless vesting
        @param _id The id of the vesting contract
    */
    function unrestrict(uint256 _id) external lock {
        address usr_ = awards[_id].usr;
        require(usr_ != address(0), "DssVest/invalid-award");
        require(wards[msg.sender] == 1 || usr_ == msg.sender, "DssVest/not-
            ↪ authorized");
        awards[_id].res = 0;
        emit Unrestrict(_id);
    }
```

19.2.32 yank(_id) [DssVest] X

```
    /**
        @dev Allows governance or the manager to remove a vesting contract
            ↪ immediately
        @param _id The id of the vesting contract
    */
    function yank(uint256 _id) external {
        _yank(_id, block.timestamp);
    }
```

19.2.33 yank(_id, _end) [DssVest] X

```
    /**
        @dev Allows governance or the manager to remove a vesting contract at a
            ↪ future time
        @param _id  The id of the vesting contract
        @param _end A scheduled time to end the vest
    */
    function yank(uint256 _id, uint256 _end) external {
        _yank(_id, _end);
    }
```

19.2.34 _yank(_id, _end) [DssVest]

```
    /**
        @dev Allows governance or the manager to end pre-maturely a vesting
            ↪ contract
        @param _id  The id of the vesting contract
        @param _end A scheduled time to end the vest
    */
    function _yank(uint256 _id, uint256 _end) internal lock {
        require(wards[msg.sender] == 1 || awards[_id].mgr == msg.sender, "
            ↪ DssVest/not-authorized");
        Award memory _award = awards[_id];
        require(_award.usr != address(0), "DssVest/invalid-award");
        if (_end < block.timestamp) {
            _end = block.timestamp;
        }
        if (_end < _award.fin) {
            uint48 end = toUint48(_end);
            awards[_id].fin = end;
            if (end < _award.bgn) {
                awards[_id].bgn = end;
                awards[_id].clf = end;
                awards[_id].tot = 0;
            } else if (end < _award.clf) {
                awards[_id].clf = end;
                awards[_id].tot = 0;
            } else {
                awards[_id].tot = toUint128(
                                    add(
                                        unpaid(_end, _award.bgn, _award.clf,
                                            ↪ _award.fin, _award.tot, _award.
                                            ↪ rxd),
                                        _award.rxd
                                    )
                                );
            }
        }

        emit Yank(_id, _end);
    }
```

19.2.35 move(_id, _dst) [DssVest] X

```
    /**
        @dev Allows owner to move a contract to a different address
        @param _id  The id of the vesting contract
        @param _dst The address to send ownership of the contract to
    */
    function move(uint256 _id, address _dst) external lock {
        require(awards[_id].usr == msg.sender, "DssVest/only-user-can-move");
        require(_dst != address(0), "DssVest/zero-address-invalid");
        awards[_id].usr = _dst;
        emit Move(_id, _dst);
    }
```

19.2.36 valid(_id) [DssVest]

```
    /**
        @dev Return true if a contract is valid
        @param _id The id of the vesting contract
        @return isValid True for valid contract
    */
    function valid(uint256 _id) external view returns (bool isValid) {
        isValid = awards[_id].rxd < awards[_id].tot;
    }
```

19.2.37 constructor(_chainlog) X

```
    /**
        @dev This contract must be authorized to 'suck' on the vat
        @param _chainlog The contract address of the MCD chainlog
    */
    constructor(address _chainlog) public DssVest() {
        require(_chainlog != address(0), "DssVestSuckable/Invalid-chainlog-
            ↪ address");
        ChainlogLike chainlog_ = chainlog = ChainlogLike(_chainlog);
        VatLike vat_ = vat = VatLike(chainlog_.getAddress("MCD_VAT"));
        DaiJoinLike daiJoin_ = daiJoin = DaiJoinLike(chainlog_.getAddress("
            ↪ MCD_JOIN_DAI"));
        vat_.hope(address(daiJoin_));
    }
```

19.2.38 pay(_guy, _amt)

```
    /**
        @dev Override pay to handle suck logic
        @param _guy The recipient of the ERC-20 Dai
        @param _amt The amount of Dai to send to the _guy [WAD]
    */
    function pay(address _guy, uint256 _amt) override internal {
        require(vat.live() == 1, "DssVestSuckable/vat-not-live");
        vat.suck(chainlog.getAddress("MCD_VOW"), address(this), mul(_amt, RAY));
        daiJoin.exit(_guy, _amt);
    }
```

19.3 contract DssVestTransferrable

```
/*
    Transferrable token DssVest. Can be used to enable streaming payments of
     any arbitrary token from an address (i.e. CU multisig) to individual
     contributors.
*/
contract DssVestTransferrable is DssVest {

    address   public immutable czar;
    TokenLike public immutable gem;
}
```

Inherited:

```
abstract contract DssVest {
    // --- Data ---
    mapping (address => uint256) public wards;
    mapping (uint256 => Award) public awards;

    uint256 public cap; // Maximum per-second issuance token rate

    uint256 public ids; // Total vestings

    uint256 internal locked;

    uint256 public constant  TWENTY_YEARS = 20 * 365 days;

    // --- Events ---
    event Rely(address indexed usr);
    event Deny(address indexed usr);

    event File(bytes32 indexed what, uint256 data);

    event Init(uint256 indexed id, address indexed usr);
    event Vest(uint256 indexed id, uint256 amt);
    event Restrict(uint256 indexed id);
    event Unrestrict(uint256 indexed id);
    event Yank(uint256 indexed id, uint256 end);
    event Move(uint256 indexed id, address indexed dst);

    /**
        @dev Override this to implement payment logic.
        @param _guy The payment target.
        @param _amt The payment amount. [units are implementation-specific]
    */
    function pay(address _guy, uint256 _amt) virtual internal;
}
```

19.3.1 struct DssVest.Award

```
    struct Award {
        address usr;   // Vesting recipient
        uint48  bgn;   // Start of vesting period  [timestamp]
        uint48  clf;   // The cliff date           [timestamp]
        uint48  fin;   // End of vesting period    [timestamp]
        address mgr;   // A manager address that can yank
        uint8   res;   // Restricted
        uint128 tot;   // Total reward amount
        uint128 rxd;   // Amount of vest claimed
    }
```

19.3.2 modifier lock() [DssVest]

```
    // --- Mutex ---
    modifier lock {
        require(locked == 0, "DssVest/system-locked");
        locked = 1;
        _;
        locked = 0;
    }
```

19.3.3 modifier auth() [DssVest]

```
    // --- Auth ---
    modifier auth {
        require(wards[msg.sender] == 1, "DssVest/not-authorized");
        _;
    }
```

19.3.4 usr(_id) [DssVest]

```
    // Getters to access only to the value desired
    function usr(uint256 _id) external view returns (address) {
        return awards[_id].usr;
    }
```

19.3.5 bgn(_id) [DssVest]

```
    function bgn(uint256 _id) external view returns (uint256) {
        return awards[_id].bgn;
    }
```

19.3.6 clf(_id) [DssVest]

```
    function clf(uint256 _id) external view returns (uint256) {
        return awards[_id].clf;
    }
```

19.3.7 fin(_id) [DssVest]

```
    function fin(uint256 _id) external view returns (uint256) {
        return awards[_id].fin;
    }
```

19.3.8 mgr(_id) [DssVest]

```
    function mgr(uint256 _id) external view returns (address) {
        return awards[_id].mgr;
    }
```

19.3.9 res(_id) [DssVest]

```
    function res(uint256 _id) external view returns (uint256) {
        return awards[_id].res;
    }
```

19.3.10 tot(_id) [DssVest]

```
    function tot(uint256 _id) external view returns (uint256) {
        return awards[_id].tot;
    }
```

19.3.11 rxd(_id) [DssVest]

```
    function rxd(uint256 _id) external view returns (uint256) {
        return awards[_id].rxd;
    }
```

19.3.12 constructor() [DssVest] X

```
    /**
        @dev Base vesting logic contract constructor
    */
    constructor() public {
        wards[msg.sender] = 1;
        emit Rely(msg.sender);
    }
```

19.3.13 rely(_usr) [DssVest] X a

```
    function rely(address _usr) external auth {
        wards[_usr] = 1;
        emit Rely(_usr);
    }
```

19.3.14 deny(_usr) [DssVest] X a

```
    function deny(address _usr) external auth {
        wards[_usr] = 0;
        emit Deny(_usr);
    }
```

19.3.15 file(what, data) [DssVest] X a

```
    /**
        @dev (Required) Set the per-second token issuance rate.
        @param what  The tag of the value to change (ex. bytes32("cap"))
        @param data  The value to update (ex. cap of 1000 tokens/yr == 1000*WAD
            ↪ /365 days)
    */
    function file(bytes32 what, uint256 data) external lock auth {
        if      (what == "cap")         cap = data;     // The maximum amount of
            ↪  tokens that can be streamed per-second per vest
        else revert("DssVest/file-unrecognized-param");
        emit File(what, data);
    }
```

19.3.16 min(x, y) [DssVest]

```
    function min(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = x > y ? y : x;
    }
```

19.3.17 add(x, y) [DssVest]

```
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x, "DssVest/add-overflow");
    }
```

19.3.18 sub(x, y) [DssVest]

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x, "DssVest/sub-underflow");
    }
```

19.3.19 mul(x, y) [DssVest]

```
    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require(y == 0 || (z = x * y) / y == x, "DssVest/mul-overflow");
    }
```

19.3.20 toUint48(x) [DssVest]

```
    function toUint48(uint256 x) internal pure returns (uint48 z) {
        require((z = uint48(x)) == x, "DssVest/uint48-overflow");
    }
```

19.3.21 toUint128(x) [DssVest]

```
    function toUint128(uint256 x) internal pure returns (uint128 z) {
        require((z = uint128(x)) == x, "DssVest/uint128-overflow");
    }
```

19.3.22 create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a

```
    /**
        @dev Govanance adds a vesting contract
        @param _usr The recipient of the reward
        @param _tot The total amount of the vest
        @param _bgn The starting timestamp of the vest
        @param _tau The duration of the vest (in seconds)
        @param _eta The cliff duration in seconds (i.e. 1 years)
        @param _mgr An optional manager for the contract. Can yank if vesting
            ↪ ends prematurely.
        @return id  The id of the vesting contract
    */
    function create(address _usr, uint256 _tot, uint256 _bgn, uint256 _tau,
       ↪ uint256 _eta, address _mgr) external lock auth returns (uint256 id) {
        require(_usr != address(0),                        "DssVest/invalid-user
            ↪ ");
        require(_tot > 0,                                  "DssVest/no-vest-
            ↪ total-amount");
        require(_bgn < add(block.timestamp, TWENTY_YEARS), "DssVest/bgn-too-far"
            ↪ );
        require(_bgn > sub(block.timestamp, TWENTY_YEARS), "DssVest/bgn-too-long
            ↪ -ago");
        require(_tau > 0,                                  "DssVest/tau-zero");
        require(_tot / _tau <= cap,                        "DssVest/rate-too-
            ↪ high");
        require(_tau <= TWENTY_YEARS,                      "DssVest/tau-too-long
            ↪ ");
        require(_eta <= _tau,                              "DssVest/eta-too-long
            ↪ ");
        require(ids < type(uint256).max,                    "DssVest/ids-overflow
            ↪ ");

        id = ++ids;
        awards[id] = Award({
            usr: _usr,
            bgn: toUint48(_bgn),
            clf: toUint48(add(_bgn, _eta)),
            fin: toUint48(add(_bgn, _tau)),
            tot: toUint128(_tot),
            rxd: 0,
            mgr: _mgr,
            res: 0
        });
        emit Init(id, _usr);
    }
```

19.3.23 vest(_id) [DssVest] X

```
    /**
        @dev Anyone (or only owner of a vesting contract if restricted) calls
            ↪ this to claim all available rewards
        @param _id      The id of the vesting contract
    */
    function vest(uint256 _id) external {
        _vest(_id, type(uint256).max);
    }
```

19.3.24 vest(_id, _maxAmt) [DssVest] X

```
    /**
        @dev Anyone (or only owner of a vesting contract if restricted) calls
            ↪ this to claim rewards
        @param _id      The id of the vesting contract
        @param _maxAmt The maximum amount to vest
    */
    function vest(uint256 _id, uint256 _maxAmt) external {
        _vest(_id, _maxAmt);
    }
```

19.3.25 _vest(_id, _maxAmt) [DssVest]

```
    /**
        @dev Anyone (or only owner of a vesting contract if restricted) calls
            ↪ this to claim rewards
        @param _id      The id of the vesting contract
        @param _maxAmt The maximum amount to vest
    */
    function _vest(uint256 _id, uint256 _maxAmt) internal lock {
        Award memory _award = awards[_id];
        require(_award.usr != address(0), "DssVest/invalid-award");
        require(_award.res == 0 || _award.usr == msg.sender, "DssVest/only-user-
            ↪ can-claim");
        uint256 amt = unpaid(block.timestamp, _award.bgn, _award.clf, _award.fin
            ↪ , _award.tot, _award.rxd);
        amt = min(amt, _maxAmt);
        awards[_id].rxd = toUint128(add(_award.rxd, amt));
        pay(_award.usr, amt);
        emit Vest(_id, amt);
    }
```

19.3.26 accrued(_id) [DssVest]

```
    /**
        @dev amount of tokens accrued, not accounting for tokens paid
        @param _id  The id of the vesting contract
        @return amt The accrued amount
    */
    function accrued(uint256 _id) external view returns (uint256 amt) {
        Award memory _award = awards[_id];
        require(_award.usr != address(0), "DssVest/invalid-award");
        amt = accrued(block.timestamp, _award.bgn, _award.fin, _award.tot);
    }
```

19.3.27 accrued(_time, _bgn, _fin, _tot) [DssVest]

```
    /**
        @dev amount of tokens accrued, not accounting for tokens paid
        @param _time The timestamp to perform the calculation
        @param _bgn  The start time of the contract
        @param _fin  The end time of the contract
        @param _tot  The total amount of the contract
        @return amt  The accrued amount
    */
    function accrued(uint256 _time, uint48 _bgn, uint48 _fin, uint128 _tot)
       ↪ internal pure returns (uint256 amt) {
        if (_time < _bgn) {
            amt = 0;
        } else if (_time >= _fin) {
            amt = _tot;
        } else {
            amt = mul(_tot, sub(_time, _bgn)) / sub(_fin, _bgn); // 0 <= amt <
               ↪ _award.tot
        }
    }
```

19.3.28 unpaid(_id) [DssVest]

```
    /**
        @dev return the amount of vested, claimable GEM for a given ID
        @param _id  The id of the vesting contract
        @return amt The claimable amount
    */
    function unpaid(uint256 _id) external view returns (uint256 amt) {
        Award memory _award = awards[_id];
        require(_award.usr != address(0), "DssVest/invalid-award");
        amt = unpaid(block.timestamp, _award.bgn, _award.clf, _award.fin, _award
           ↪ .tot, _award.rxd);
    }
```

19.3.29 unpaid(_time, _bgn, _clf, _fin, _tot, _rxd) [DssVest]

```
    /**
        @dev amount of tokens accrued, not accounting for tokens paid
        @param _time The timestamp to perform the calculation
        @param _bgn  The start time of the contract
        @param _clf  The timestamp of the cliff
        @param _fin  The end time of the contract
        @param _tot  The total amount of the contract
        @param _rxd  The number of gems received
        @return amt  The claimable amount
    */
    function unpaid(uint256 _time, uint48 _bgn, uint48 _clf, uint48 _fin,
       ↪ uint128 _tot, uint128 _rxd) internal pure returns (uint256 amt) {
        amt = _time < _clf ? 0 : sub(accrued(_time, _bgn, _fin, _tot), _rxd);
    }
```

19.3.30 restrict(_id) [DssVest] X

```
    /**
        @dev Allows governance or the owner to restrict vesting to the owner
            ↪ only
        @param _id The id of the vesting contract
    */
    function restrict(uint256 _id) external lock {
        address usr_ = awards[_id].usr;
        require(usr_ != address(0), "DssVest/invalid-award");
        require(wards[msg.sender] == 1 || usr_ == msg.sender, "DssVest/not-
            ↪ authorized");
        awards[_id].res = 1;
        emit Restrict(_id);
    }
```

19.3.31 unrestrict(_id) [DssVest] X

```
    /**
        @dev Allows governance or the owner to enable permissionless vesting
        @param _id The id of the vesting contract
    */
    function unrestrict(uint256 _id) external lock {
        address usr_ = awards[_id].usr;
        require(usr_ != address(0), "DssVest/invalid-award");
        require(wards[msg.sender] == 1 || usr_ == msg.sender, "DssVest/not-
            ↪ authorized");
        awards[_id].res = 0;
        emit Unrestrict(_id);
    }
```

19.3.32 yank(_id) [DssVest] X

```
    /**
        @dev Allows governance or the manager to remove a vesting contract
            ↪ immediately
        @param _id The id of the vesting contract
    */
    function yank(uint256 _id) external {
        _yank(_id, block.timestamp);
    }
```

19.3.33 yank(_id, _end) [DssVest] X

```
    /**
        @dev Allows governance or the manager to remove a vesting contract at a
            ↪ future time
        @param _id  The id of the vesting contract
        @param _end A scheduled time to end the vest
    */
    function yank(uint256 _id, uint256 _end) external {
        _yank(_id, _end);
    }
```

19.3.34 _yank(_id, _end) [DssVest]

```
    /**
        @dev Allows governance or the manager to end pre-maturely a vesting
           ↪ contract
        @param _id  The id of the vesting contract
        @param _end A scheduled time to end the vest
    */
    function _yank(uint256 _id, uint256 _end) internal lock {
        require(wards[msg.sender] == 1 || awards[_id].mgr == msg.sender, "
           ↪ DssVest/not-authorized");
        Award memory _award = awards[_id];
        require(_award.usr != address(0), "DssVest/invalid-award");
        if (_end < block.timestamp) {
            _end = block.timestamp;
        }
        if (_end < _award.fin) {
            uint48 end = toUint48(_end);
            awards[_id].fin = end;
            if (end < _award.bgn) {
                awards[_id].bgn = end;
                awards[_id].clf = end;
                awards[_id].tot = 0;
            } else if (end < _award.clf) {
                awards[_id].clf = end;
                awards[_id].tot = 0;
            } else {
                awards[_id].tot = toUint128(
                                    add(
                                        unpaid(_end, _award.bgn, _award.clf,
                                           ↪ _award.fin, _award.tot, _award.
                                           ↪ rxd),
                                        _award.rxd
                                    )
                                );
            }
        }

        emit Yank(_id, _end);
    }
```

19.3.35 move(_id, _dst) [DssVest] X

```
    /**
        @dev Allows owner to move a contract to a different address
        @param _id  The id of the vesting contract
        @param _dst The address to send ownership of the contract to
    */
    function move(uint256 _id, address _dst) external lock {
        require(awards[_id].usr == msg.sender, "DssVest/only-user-can-move");
        require(_dst != address(0), "DssVest/zero-address-invalid");
        awards[_id].usr = _dst;
        emit Move(_id, _dst);
    }
```

19.3.36 valid(_id) [DssVest]

```
    /**
        @dev Return true if a contract is valid
        @param _id The id of the vesting contract
        @return isValid True for valid contract
    */
    function valid(uint256 _id) external view returns (bool isValid) {
        isValid = awards[_id].rxd < awards[_id].tot;
    }
```

19.3.37 constructor(_czar, _gem) X

```
/**
    @dev This contract must be approved for transfer of the gem on the czar
    @param _czar The owner of the tokens to be distributed
    @param _gem  The token to be distributed
*/
constructor(address _czar, address _gem) public DssVest() {
    require(_czar != address(0), "DssVestTransferrable/Invalid-distributor-
        ↪ address");
    require(_gem  != address(0), "DssVestTransferrable/Invalid-token-address
        ↪ ");
    czar = _czar;
    gem  = TokenLike(_gem);
}
```

19.3.38 pay(_guy, _amt)

```
/**
    @dev Override pay to handle transfer logic
    @param _guy The recipient of the ERC-20 Dai
    @param _amt The amount of gem to send to the _guy (in native token units
        ↪ )
*/
function pay(address _guy, uint256 _amt) override internal {
    require(gem.transferFrom(czar, _guy, _amt), "DssVestTransferrable/failed
        ↪ -transfer");
}
```

Chapter 20

Unassigned

20.1 contract DssAutoLine

```
contract DssAutoLine {

    mapping (bytes32 => Ilk)     public ilks;
    mapping (address => uint256) public wards;

    VatLike immutable public vat;

    /*** Events ***/
    event Rely(address indexed usr);
    event Deny(address indexed usr);
    event Setup(bytes32 indexed ilk, uint256 line, uint256 gap, uint256 ttl);
    event Remove(bytes32 indexed ilk);
    event Exec(bytes32 indexed ilk, uint256 line, uint256 lineNew);

    /*** Administration ***/
}
```

20.1.1 struct DssAutoLine.Ilk

```
    /*** Data ***/
    struct Ilk {
        uint256   line;  // Max ceiling possible
            ↪                                                [rad]
        uint256    gap;  // Max Value between current debt and line to be set
            ↪                    [rad]
        uint48     ttl;  // Min time to pass before a new increase
            ↪                              [seconds]
        uint48    last;  // Last block the ceiling was updated
            ↪                                  [blocks]
        uint48 lastInc;  // Last time the ceiling was increased compared to its
            ↪ previous value [seconds]
    }
```

20.1.2 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "DssAutoLine/not-authorized");
        _;
    }
```

20.1.3 constructor(vat_) X

```
    /*** Init ***/
    constructor(address vat_) public {
        vat = VatLike(vat_);
        wards[msg.sender] = 1;
        emit Rely(msg.sender);
    }
```

20.1.4 add(x, y)

```
    /*** Math ***/
    function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x + y) >= x);
    }
```

20.1.5 sub(x, y)

```
    function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require((z = x - y) <= x);
    }
```

20.1.6 mul(x, y)

```
    function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
        require(y == 0 || (z = x * y) / y == x);
    }
```

20.1.7 min(x, y)

```
    function min(uint256 x, uint256 y) internal pure returns (uint256 z) {
        return x <= y ? x : y;
    }
```

20.1.8 setIlk(ilk, line, gap, ttl) X a

```
    /**
        @dev Add or update an ilk
        @param ilk    Collateral type (ex. ETH-A)
        @param line   Collateral maximum debt ceiling that can be configured [
            ↪ RAD]
        @param gap    Amount of collateral to step [RAD]
        @param ttl    Minimum time between increase [seconds]
    */
    function setIlk(bytes32 ilk, uint256 line, uint256 gap, uint256 ttl)
       ↪ external auth {
        require(ttl  < uint48(-1), "DssAutoLine/invalid-ttl");
        require(line > 0,          "DssAutoLine/invalid-line");
        ilks[ilk] = Ilk(line, gap, uint48(ttl), 0, 0);
        emit Setup(ilk, line, gap, ttl);
    }
```

20.1.9 remIlk(ilk) X a

```
    /**
        @dev Remove an ilk
        @param ilk    Collateral type (ex. ETH-A)
    */
    function remIlk(bytes32 ilk) external auth {
        delete ilks[ilk];
        emit Remove(ilk);
    }
```

20.1.10 rely(usr) X a

```
    function rely(address usr) external auth {
        wards[usr] = 1;
        emit Rely(usr);
    }
```

20.1.11 deny(usr) X a

```
    function deny(address usr) external auth {
        wards[usr] = 0;
        emit Deny(usr);
    }
```

20.1.12 exec(_ilk) X

```
    /*** Auto-Line Update ***/
    // @param  _ilk  The bytes32 ilk tag to adjust (ex. "ETH-A")
    // @return       The ilk line value as uint256
    function exec(bytes32 _ilk) external returns (uint256) {
        (uint256 Art, uint256 rate,, uint256 line,) = vat.ilks(_ilk);
        uint256 ilkLine = ilks[_ilk].line;

        // Return if the ilk is not enabled
        if (ilkLine == 0) return line;

        // 1 SLOAD
        uint48 ilkTtl     = ilks[_ilk].ttl;
        uint48 ilkLast    = ilks[_ilk].last;
        uint48 ilkLastInc = ilks[_ilk].lastInc;
        //

        // Return if there was already an update in the same block
        if (ilkLast == block.number) return line;

        // Calculate collateral debt
        uint256 debt = mul(Art, rate);

        uint256 ilkGap  = ilks[_ilk].gap;

        // Calculate new line based on the minimum between the maximum line and
            ↪ actual collateral debt + gap
        uint256 lineNew = min(add(debt, ilkGap), ilkLine);

        // Short-circuit if there wasn't an update or if the time since last
            ↪ increment has not passed
        if (lineNew == line || lineNew > line && block.timestamp < add(
            ↪ ilkLastInc, ilkTtl)) return line;

        // Set collateral debt ceiling
        vat.file(_ilk, "line", lineNew);
        // Set general debt ceiling
        vat.file("Line", add(sub(vat.Line(), line), lineNew));

        // Update lastInc if it is an increment in the debt ceiling
        // and update last whatever the update is
        if (lineNew > line) {
            // 1 SSTORE
            ilks[_ilk].lastInc = uint48(block.timestamp);
            ilks[_ilk].last    = uint48(block.number);
            //
        } else {
            ilks[_ilk].last    = uint48(block.number);
        }

        emit Exec(_ilk, line, lineNew);
        return lineNew;
    }
```

20.2 contract ChainLog

```
/// @title An on-chain governance-managed contract registry
/// @notice Publicly readable data; mutating functions must be called by an
    ↪ authorized user
contract ChainLog {

    event Rely(address usr);
    event Deny(address usr);
    event UpdateVersion(string version);
    event UpdateSha256sum(string sha256sum);
    event UpdateIPFS(string ipfs);
    event UpdateAddress(bytes32 key, address addr);
    event RemoveAddress(bytes32 key);

    // --- Auth ---
    mapping (address => uint) public wards;
    mapping (bytes32 => Location) location;

    bytes32[] public keys;

    string public version;
    string public sha256sum;
    string public ipfs;
}
```

20.2.1 struct ChainLog.Location

```
    struct Location {
        uint256  pos;
        address  addr;
    }
```

20.2.2 modifier auth()

```
    modifier auth {
        require(wards[msg.sender] == 1, "ChainLog/not-authorized");
        _;
    }
```

20.2.3 rely(usr) X a

```
    function rely(address usr) external auth { wards[usr] = 1; emit Rely(usr); }
```

20.2.4 deny(usr) X a

```
    function deny(address usr) external auth { wards[usr] = 0; emit Deny(usr); }
```

20.2.5 constructor() X

```
    constructor() public {
        wards[msg.sender] = 1;
        setVersion("0.0.0");
        setAddress("CHANGELOG", address(this));
    }
```

20.2.6 setVersion(_version) X a

```
    /// @notice Set the "version" of the current changelog
    /// @param _version The version string (optional)
    function setVersion(string memory _version) public auth {
        version = _version;
        emit UpdateVersion(_version);
    }
```

20.2.7 setSha256sum(_sha256sum) X a

```
    /// @notice Set the "sha256sum" of some current external changelog
    /// @dev designed to store sha256 of changelog.makerdao.com hosted log
    /// @param _sha256sum The sha256 sum (optional)
    function setSha256sum(string memory _sha256sum) public auth {
        sha256sum = _sha256sum;
        emit UpdateSha256sum(_sha256sum);
    }
```

20.2.8 setIPFS(_ipfs) X a

```
    /// @notice Set the IPFS hash of a pinned changelog
    /// @dev designed to store IPFS pin hash that can retreive changelog json
    /// @param _ipfs The ipfs pin hash of an ipfs hosted log (optional)
    function setIPFS(string memory _ipfs) public auth {
        ipfs = _ipfs;
        emit UpdateIPFS(_ipfs);
    }
```

20.2.9 setAddress(_key, _addr) X a

```
    /// @notice Set the key-value pair for a changelog item
    /// @param _key  the changelog key (ex. MCD_VAT)
    /// @param _addr the address to the contract
    function setAddress(bytes32 _key, address _addr) public auth {
        if (count() > 0 && _key == keys[location[_key].pos]) {
            location[_key].addr = _addr;   // Key exists in keys (update)
        } else {
            _addAddress(_key, _addr);      // Add key to keys array
        }
        emit UpdateAddress(_key, _addr);
    }
```

20.2.10 removeAddress(_key) X a

```
    /// @notice Removes the key from the keys list()
    /// @dev removes the item from the array but moves the last element to it's
      ↪ place
    //   WARNING: To save the expense of shifting an array on-chain,
    //     this will replace the key to be deleted with the last key
    //     in the array, and can therefore result in keys being out
    //     of order. Use this only if you intend to reorder the list(),
    //     otherwise consider using `setAddress("KEY", address(0));`
    /// @param _key the key to be removed
    function removeAddress(bytes32 _key) public auth {
        _removeAddress(_key);
        emit RemoveAddress(_key);
    }
```

20.2.11 count()

```
/// @notice Returns the number of keys being tracked in the keys array
/// @return the number of keys as uint256
function count() public view returns (uint256) {
    return keys.length;
}
```

20.2.12 get(_index)

```
/// @notice Returns the key and address of an item in the changelog array (
    ↪ for enumeration)
/// @dev _index is 0-indexed to the underlying array
/// @return a tuple containing the key and address associated with that key
function get(uint256 _index) public view returns (bytes32, address) {
    return (keys[_index], location[keys[_index]].addr);
}
```

20.2.13 list()

```
/// @notice Returns the list of keys being tracked by the changelog
/// @dev May fail if keys is too large, if so, call count() and iterate with
    ↪  get()
function list() public view returns (bytes32[] memory) {
    return keys;
}
```

20.2.14 getAddress(_key)

```
/// @notice Returns the address for a particular key
/// @param _key a bytes32 key (ex. MCD_VAT)
/// @return addr the contract address associated with the key
function getAddress(bytes32 _key) public view returns (address addr) {
    addr = location[_key].addr;
    require(addr != address(0), "dss-chain-log/invalid-key");
}
```

20.2.15 _addAddress(_key, _addr)

```
function _addAddress(bytes32 _key, address _addr) internal {
    keys.push(_key);
    location[keys[keys.length - 1]] = Location(
        keys.length - 1,
        _addr
    );
}
```

20.2.16 _removeAddress(_key)

```
function _removeAddress(bytes32 _key) internal {
    uint256 index = location[_key].pos;       // Get pos in array
    require(keys[index] == _key, "dss-chain-log/invalid-key");
    bytes32 move  = keys[keys.length - 1];    // Get last key
    keys[index] = move;                       // Replace
    location[move].pos = index;               // Update array pos
    keys.pop();                               // Trim last key
    delete location[_key];                    // Delete struct data
}
```

Chapter 21

Dependencies

21.1 library ScriptTools

```
/**
 * @title Script Tools
 * @dev Contains opinionated tools used in scripts.
 */
library ScriptTools {

    VmSafe private constant vm = VmSafe(address(uint160(uint256(keccak256("hevm
       ↪ cheat code")))));

    string internal constant DEFAULT_DELIMITER = ",";
    string internal constant DELIMITER_OVERRIDE = "DSSTEST_ARRAY_DELIMITER";
    string internal constant EXPORT_JSON_KEY = "EXPORTS";

}
```

21.1.1 getRootChainId()

```
    function getRootChainId() internal view returns (uint256) {
        return vm.envUint("FOUNDRY_ROOT_CHAINID");
    }
```

21.1.2 readInput(name)

```
    function readInput(string memory name) internal view returns (string memory)
       ↪ {
        string memory root = vm.projectRoot();
        string memory chainInputFolder = string(abi.encodePacked("/script/input/
           ↪ ", vm.toString(getRootChainId()), "/"));
        return vm.readFile(string(abi.encodePacked(root, chainInputFolder, name,
           ↪ ".json")));
    }
```

21.1.3 readOutput(name, timestamp)

```
    function readOutput(string memory name, uint256 timestamp) internal view
       ↪ returns (string memory) {
        string memory root = vm.projectRoot();
        string memory chainOutputFolder = string(abi.encodePacked("/script/
           ↪ output/", vm.toString(getRootChainId()), "/"));
        return vm.readFile(string(abi.encodePacked(root, chainOutputFolder, name
           ↪ , "-", vm.toString(timestamp), ".json")));
    }
```

21.1.4 readOutput(name)

```
function readOutput(string memory name) internal view returns (string memory
    ↪ ) {
    string memory root = vm.projectRoot();
    string memory chainOutputFolder = string(abi.encodePacked("/script/
        ↪ output/", vm.toString(getRootChainId()), "/"));
    return vm.readFile(string(abi.encodePacked(root, chainOutputFolder, name
        ↪ , "-latest.json")));
}
```

21.1.5 loadConfig(name)

```
/**
 * @notice Use standard environment variables to load config.
 * @dev Will first check FOUNDRY_SCRIPT_CONFIG_TEXT for raw json text.
 *      Falls back to FOUNDRY_SCRIPT_CONFIG for a standard file definition.
 *      Finally will fall back to the given string `name`.
 * @param name The default config file to load if no environment variables
     ↪ are set.
 * @return config The raw json text of the config.
 */
function loadConfig(string memory name) internal returns (string memory
    ↪ config) {
    config = vm.envOr("FOUNDRY_SCRIPT_CONFIG_TEXT", string(""));
    if (eq(config, "")) {
        config = readInput(vm.envOr("FOUNDRY_SCRIPT_CONFIG", name));
    }
}
```

21.1.6 loadConfig()

```
/**
 * @notice Use standard environment variables to load config.
 * @dev Will first check FOUNDRY_SCRIPT_CONFIG_TEXT for raw json text.
 *      Falls back to FOUNDRY_SCRIPT_CONFIG for a standard file definition.
 *      Finally will revert if no environment variables are set.
 * @return config The raw json text of the config.
 */
function loadConfig() internal returns (string memory config) {
    config = vm.envOr("FOUNDRY_SCRIPT_CONFIG_TEXT", string(""));
    if (eq(config, "")) {
        config = readInput(vm.envString("FOUNDRY_SCRIPT_CONFIG"));
    }
}
```

21.1.7 loadDependencies(name)

```
/**
 * @notice Use standard environment variables to load dependencies.
 * @dev Will first check FOUNDRY_SCRIPT_DEPS_TEXT for raw json text.
 *      Falls back to FOUNDRY_SCRIPT_DEPS for a standard file definition.
 *      Finally will fall back to the given string `name`.
 * @param name The default dependency file to load if no environment
     ↪ variables are set.
 * @return dependencies The raw json text of the dependencies.
 */
function loadDependencies(string memory name) internal returns (string
    ↪ memory dependencies) {
    dependencies = vm.envOr("FOUNDRY_SCRIPT_DEPS_TEXT", string(""));
    if (eq(dependencies, "")) {
        dependencies = readOutput(vm.envOr("FOUNDRY_SCRIPT_DEPS", name));
    }
}
```

21.1.8 loadDependencies()

```
/**
 * @notice Use standard environment variables to load dependencies.
 * @dev Will first check FOUNDRY_SCRIPT_DEPS_TEXT for raw json text.
 *      Falls back to FOUNDRY_SCRIPT_DEPS for a standard file definition.
 *      Finally will revert if no environment variables are set.
 * @return dependencies The raw json text of the dependencies.
 */
function loadDependencies() internal returns (string memory dependencies) {
    dependencies = vm.envOr("FOUNDRY_SCRIPT_DEPS_TEXT", string(""));
    if (eq(dependencies, "")) {
        dependencies = readOutput(vm.envString("FOUNDRY_SCRIPT_DEPS"));
    }
}
```

21.1.9 exportContract(name, label, addr)

```
/**
 * @notice Used to export important contracts to higher level deploy scripts
    ↪ .
 *         Note waiting on Foundry to have better primitives, but roll our
    ↪ own for now.
 * @dev Set FOUNDRY_EXPORTS_NAME to override the name of the json file.
 * @param name The name to give the json file.
 * @param label The label of the address.
 * @param addr The address to export.
 */
function exportContract(string memory name, string memory label, address
   ↪ addr) internal {
    name = vm.envOr("FOUNDRY_EXPORTS_NAME", name);
    string memory json = vm.serializeAddress(EXPORT_JSON_KEY, label, addr);
    string memory root = vm.projectRoot();
    string memory chainOutputFolder = string(abi.encodePacked("/script/
        ↪ output/", vm.toString(getRootChainId()), "/"));
    vm.writeJson(json, string(abi.encodePacked(root, chainOutputFolder, name
        ↪ , "-", vm.toString(block.timestamp), ".json")));
    if (vm.envOr("FOUNDRY_EXPORTS_OVERWRITE_LATEST", false)) {
        vm.writeJson(json, string(abi.encodePacked(root, chainOutputFolder,
            ↪ name, "-latest.json")));
    }
}
```

21.1.10 exportContract(label, addr)

```
/**
 * @notice Used to export important contracts to higher level deploy scripts
    ↪ .
 *         Note waiting on Foundry to have better primitives, but roll our
    ↪ own for now.
 * @dev Requires FOUNDRY_EXPORTS_NAME to be set.
 * @param label The label of the address.
 * @param addr The address to export.
 */
function exportContract(string memory label, address addr) internal {
    exportContract(vm.envString("FOUNDRY_EXPORTS_NAME"), label, addr);
}
```

21.1.11 stringToBytes32(source)

```
/**
 * @notice It's common to define strings as bytes32 (such as for ilks)
 */
```

```
    function stringToBytes32(string memory source) internal pure returns (
        ↪ bytes32 result) {
        bytes memory emptyStringTest = bytes(source);
        if (emptyStringTest.length == 0) {
            return 0x0;
        }

        assembly {
            result := mload(add(source, 32))
        }
    }
```

21.1.12 ilkToChainlogFormat(ilk)

```
    /**
     * @notice Convert an ilk to a chainlog key by replacing all dashes with
         ↪ underscores.
     *          Ex) Convert "ETH-A" to "ETH_A"
     */
    function ilkToChainlogFormat(bytes32 ilk) internal pure returns (string
        ↪ memory) {
        uint256 len = 0;
        for (; len < 32; len++) {
            if (uint8(ilk[len]) == 0x00) break;
        }
        bytes memory result = new bytes(len);
        for (uint256 i = 0; i < len; i++) {
            uint8 b = uint8(ilk[i]);
            if (b == 0x2d) result[i] = bytes1(0x5f);
            else result[i] = bytes1(b);
        }
        return string(result);
    }
```

21.1.13 eq(a, b)

```
    function eq(string memory a, string memory b) internal pure returns (bool) {
        return keccak256(bytes(a)) == keccak256(bytes(b));
    }
```

21.1.14 switchOwner(base, deployer, newOwner)

```
    function switchOwner(address base, address deployer, address newOwner)
        ↪ internal {
        if (deployer == newOwner) return;
        require(WardsAbstract(base).wards(deployer) == 1, "deployer-not-authed")
            ↪ ;
        WardsAbstract(base).rely(newOwner);
        WardsAbstract(base).deny(deployer);
    }
```

21.1.15 readUint(json, key, envKey)

```
    // Read config variable, but allow for an environment variable override

    function readUint(string memory json, string memory key, string memory
        ↪ envKey) internal returns (uint256) {
        return vm.envOr(envKey, stdJson.readUint(json, key));
    }
```

21.1.16 readUintArray(json, key, envKey)

```
function readUintArray(string memory json, string memory key, string memory
    ↪ envKey) internal returns (uint256[] memory) {
    return vm.envOr(envKey, vm.envOr(DELIMITER_OVERRIDE, DEFAULT_DELIMITER),
        ↪ stdJson.readUintArray(json, key));
}
```

21.1.17 readInt(json, key, envKey)

```
function readInt(string memory json, string memory key, string memory envKey
    ↪ ) internal returns (int256) {
    return vm.envOr(envKey, stdJson.readInt(json, key));
}
```

21.1.18 readIntArray(json, key, envKey)

```
function readIntArray(string memory json, string memory key, string memory
    ↪ envKey) internal returns (int256[] memory) {
    return vm.envOr(envKey, vm.envOr(DELIMITER_OVERRIDE, DEFAULT_DELIMITER),
        ↪ stdJson.readIntArray(json, key));
}
```

21.1.19 readBytes32(json, key, envKey)

```
function readBytes32(string memory json, string memory key, string memory
    ↪ envKey) internal returns (bytes32) {
    return vm.envOr(envKey, stdJson.readBytes32(json, key));
}
```

21.1.20 readBytes32Array(json, key, envKey)

```
function readBytes32Array(string memory json, string memory key, string
    ↪ memory envKey) internal returns (bytes32[] memory) {
    return vm.envOr(envKey, vm.envOr(DELIMITER_OVERRIDE, DEFAULT_DELIMITER),
        ↪ stdJson.readBytes32Array(json, key));
}
```

21.1.21 readString(json, key, envKey)

```
function readString(string memory json, string memory key, string memory
    ↪ envKey) internal returns (string memory) {
    return vm.envOr(envKey, stdJson.readString(json, key));
}
```

21.1.22 readStringArray(json, key, envKey)

```
function readStringArray(string memory json, string memory key, string
    ↪ memory envKey) internal returns (string[] memory) {
    return vm.envOr(envKey, vm.envOr(DELIMITER_OVERRIDE, DEFAULT_DELIMITER),
        ↪ stdJson.readStringArray(json, key));
}
```

21.1.23 readAddress(json, key, envKey)

```
function readAddress(string memory json, string memory key, string memory
    ↪ envKey) internal returns (address) {
    return vm.envOr(envKey, stdJson.readAddress(json, key));
}
```

21.1.24 readAddressArray(json, key, envKey)

```
function readAddressArray(string memory json, string memory key, string
    ↪ memory envKey) internal returns (address[] memory) {
    return vm.envOr(envKey, vm.envOr(DELIMITER_OVERRIDE, DEFAULT_DELIMITER),
        ↪  stdJson.readAddressArray(json, key));
}
```

21.1.25 readBool(json, key, envKey)

```
function readBool(string memory json, string memory key, string memory
    ↪ envKey) internal returns (bool) {
    return vm.envOr(envKey, stdJson.readBool(json, key));
}
```

21.1.26 readBoolArray(json, key, envKey)

```
function readBoolArray(string memory json, string memory key, string memory
    ↪ envKey) internal returns (bool[] memory) {
    return vm.envOr(envKey, vm.envOr(DELIMITER_OVERRIDE, DEFAULT_DELIMITER),
        ↪  stdJson.readBoolArray(json, key));
}
```

21.1.27 readBytes(json, key, envKey)

```
function readBytes(string memory json, string memory key, string memory
    ↪ envKey) internal returns (bytes memory) {
    return vm.envOr(envKey, stdJson.readBytes(json, key));
}
```

21.1.28 readBytesArray(json, key, envKey)

```
function readBytesArray(string memory json, string memory key, string memory
    ↪  envKey) internal returns (bytes[] memory) {
    return vm.envOr(envKey, vm.envOr(DELIMITER_OVERRIDE, DEFAULT_DELIMITER),
        ↪  stdJson.readBytesArray(json, key));
}
```

21.2 library MCD

```
library MCD {

    uint256 constant WAD = 10 ** 18;
    uint256 constant RAY = 10 ** 27;
    uint256 constant RAD = 10 ** 45;

}
```

21.2.1 getAddressOrNull(dss, key)

```
    function getAddressOrNull(DssInstance memory dss, bytes32 key) internal view
        ↪  returns (address) {
        try dss.chainlog.getAddress(key) returns (address a) {
            return a;
        } catch {
            return address(0);
        }
    }
```

21.2.2 loadFromChainlog(chainlog)

```
    function loadFromChainlog(address chainlog) internal view returns (
        ↪ DssInstance memory dss) {
        return loadFromChainlog(ChainlogAbstract(chainlog));
    }
```

21.2.3 loadFromChainlog(chainlog)

```
    function loadFromChainlog(ChainlogAbstract chainlog) internal view returns (
        ↪ DssInstance memory dss) {
        dss.chainlog = chainlog;
        dss.vat = VatAbstract(getAddressOrNull(dss, "MCD_VAT"));
        dss.daiJoin = DaiJoinAbstract(getAddressOrNull(dss, "MCD_JOIN_DAI"));
        dss.dai = DaiAbstract(getAddressOrNull(dss, "MCD_DAI"));
        dss.vow = VowAbstract(getAddressOrNull(dss, "MCD_VOW"));
        dss.dog = DogAbstract(getAddressOrNull(dss, "MCD_DOG"));
        dss.pot = PotAbstract(getAddressOrNull(dss, "MCD_POT"));
        dss.jug = JugAbstract(getAddressOrNull(dss, "MCD_JUG"));
        dss.spotter = SpotAbstract(getAddressOrNull(dss, "MCD_SPOT"));
        dss.end = EndAbstract(getAddressOrNull(dss, "MCD_END"));
        dss.cure = CureAbstract(getAddressOrNull(dss, "MCD_CURE"));
        dss.flap = FlapAbstract(getAddressOrNull(dss, "MCD_FLAP"));
        dss.flop = FlopAbstract(getAddressOrNull(dss, "MCD_FLOP"));
        dss.esm = ESMAbstract(getAddressOrNull(dss, "MCD_ESM"));
    }
```

21.2.4 bytesToBytes32(b)

```
    function bytesToBytes32(bytes memory b) private pure returns (bytes32) {
        bytes32 out;
        for (uint256 i = 0; i < b.length; i++) {
            out |= bytes32(b[i] & 0xFF) >> (i * 8);
        }
        return out;
    }
```

21.2.5 getIlk(dss, gem, variant)

```
    function getIlk(DssInstance memory dss, string memory gem, string memory
        ↪ variant) internal view returns (DssIlkInstance memory) {
        return DssIlkInstance(
            DSTokenAbstract(getAddressOrNull(dss, bytesToBytes32(bytes(gem)))),
            OsmAbstract(getAddressOrNull(dss, bytesToBytes32(abi.encodePacked("
                ↪ PIP_", gem)))),
            GemJoinAbstract(getAddressOrNull(dss, bytesToBytes32(abi.
                ↪ encodePacked("MCD_JOIN_", gem, "_", variant)))),
            ClipAbstract(getAddressOrNull(dss, bytesToBytes32(abi.encodePacked("
                ↪ MCD_CLIP_", gem, "_", variant))))
        );
    }
```

21.2.6 initIlk(dss, ilk)

```
    /// @dev Initialize a dummy ilk with a $1 DSValue pip without liquidations
    function initIlk(
        DssInstance memory dss,
        bytes32 ilk
    ) internal {
        DSValue pip = new DSValue();
        pip.poke(bytes32(WAD));
        initIlk(dss, ilk, address(0), address(pip));
    }
```

21.2.7 initIlk(dss, ilk, join)

```
    /// @dev Initialize an ilk with a $1 DSValue pip without liquidations
    function initIlk(
        DssInstance memory dss,
        bytes32 ilk,
        address join
    ) internal {
        DSValue pip = new DSValue();
        pip.poke(bytes32(WAD));
        initIlk(dss, ilk, join, address(pip));
    }
```

21.2.8 initIlk(dss, ilk, join, pip)

```
    /// @dev Initialize an ilk without liquidations
    function initIlk(
        DssInstance memory dss,
        bytes32 ilk,
        address join,
        address pip
    ) internal {
        dss.vat.init(ilk);
        dss.jug.init(ilk);

        dss.vat.rely(join);

        dss.spotter.file(ilk, "pip", pip);
        dss.spotter.file(ilk, "mat", RAY);
        dss.spotter.poke(ilk);
    }
```

21.2.9 initIlk(dss, ilk, join, pip, clip, clipCalc)

```
    /// @dev Initialize an ilk with liquidations
    function initIlk(
        DssInstance memory dss,
        bytes32 ilk,
        address join,
        address pip,
        address clip,
        address clipCalc
    ) internal {
        initIlk(dss, ilk, join, pip);

        // TODO liquidations
        clip; clipCalc;
    }
```

21.2.10 giveAdminAccess(dss, who)

```
    /// @dev Give who a ward on all core contracts
    function giveAdminAccess(DssInstance memory dss, address who) internal {
        if (address(dss.vat) != address(0)) GodMode.setWard(address(dss.vat),
            ↪ who, 1);
        if (address(dss.dai) != address(0)) GodMode.setWard(address(dss.dai),
            ↪ who, 1);
        if (address(dss.vow) != address(0)) GodMode.setWard(address(dss.vow),
            ↪ who, 1);
        if (address(dss.dog) != address(0)) GodMode.setWard(address(dss.dog),
            ↪ who, 1);
        if (address(dss.pot) != address(0)) GodMode.setWard(address(dss.pot),
            ↪ who, 1);
        if (address(dss.jug) != address(0)) GodMode.setWard(address(dss.jug),
            ↪ who, 1);
        if (address(dss.spotter) != address(0)) GodMode.setWard(address(dss.
            ↪ spotter), who, 1);
        if (address(dss.end) != address(0)) GodMode.setWard(address(dss.end),
            ↪ who, 1);
        if (address(dss.cure) != address(0)) GodMode.setWard(address(dss.cure),
            ↪ who, 1);
        if (address(dss.esm) != address(0)) GodMode.setWard(address(dss.esm),
            ↪ who, 1);
    }
```

21.2.11 giveAdminAccess(dss)

```
    /// @dev Give who a ward on all core contracts to this address
    function giveAdminAccess(DssInstance memory dss) internal {
        giveAdminAccess(dss, address(this));
    }
```

21.2.12 newUser(dss)

```
    function newUser(DssInstance memory dss) internal returns (MCDUser) {
        return new MCDUser(dss);
    }
```

21.3 library D3MDeploy

```
// Deploy a D3M instance
library D3MDeploy {

}
```

21.3.1 deployCore(deployer, owner, daiJoin)

```
    function deployCore(
        address deployer,
        address owner,
        address daiJoin
    ) internal returns (D3MCoreInstance memory d3mCore) {
        d3mCore.hub = address(new D3MHub(daiJoin));
        d3mCore.mom = address(new D3MMom());

        ScriptTools.switchOwner(d3mCore.hub, deployer, owner);
        DSAuthAbstract(d3mCore.mom).setOwner(owner);
    }
```

21.3.2 deployAave(deployer, owner, ilk, vat, hub, dai, lendingPool)

```
    function deployAave(
        address deployer,
        address owner,
        bytes32 ilk,
        address vat,
        address hub,
        address dai,
        address lendingPool
    ) internal returns (D3MInstance memory d3m) {
        d3m.plan = address(new D3MAavePlan(dai, lendingPool));
        d3m.pool = address(new D3MAavePool(ilk, hub, dai, lendingPool));
        d3m.oracle = address(new D3MOracle(vat, ilk));

        ScriptTools.switchOwner(d3m.plan, deployer, owner);
        ScriptTools.switchOwner(d3m.pool, deployer, owner);
        ScriptTools.switchOwner(d3m.oracle, deployer, owner);
    }
```

21.3.3 deployCompound(deployer, owner, ilk, vat, hub, cdai)

```
    function deployCompound(
        address deployer,
        address owner,
        bytes32 ilk,
        address vat,
        address hub,
        address cdai
    ) internal returns (D3MInstance memory d3m) {
        d3m.plan = address(new D3MCompoundPlan(cdai));
        d3m.pool = address(new D3MCompoundPool(ilk, hub, cdai));
        d3m.oracle = address(new D3MOracle(vat, ilk));

        ScriptTools.switchOwner(d3m.plan, deployer, owner);
        ScriptTools.switchOwner(d3m.pool, deployer, owner);
        ScriptTools.switchOwner(d3m.oracle, deployer, owner);
    }
```

21.4 library D3MInit

```
// Init a D3M instance
library D3MInit {

}
```

21.4.1 initCore(dss, d3mCore)

```
    function initCore(
        DssInstance memory dss,
        D3MCoreInstance memory d3mCore
    ) internal {
        D3MHubLike hub = D3MHubLike(d3mCore.hub);
        D3MMomLike mom = D3MMomLike(d3mCore.mom);

        // Sanity checks
        require(hub.vat() == address(dss.vat), "Hub vat mismatch");
        require(hub.daiJoin() == address(dss.daiJoin), "Hub daiJoin mismatch");

        hub.file("vow", address(dss.vow));
        hub.file("end", address(dss.end));

        mom.setAuthority(dss.chainlog.getAddress("MCD_ADM"));

        dss.vat.rely(address(hub));

        dss.chainlog.setAddress("DIRECT_HUB", address(hub));
        dss.chainlog.setAddress("DIRECT_MOM", address(mom));
    }
```

21.4.2 _init(dss, d3m, cfg, gem)

```
    function _init(
        DssInstance memory dss,
        D3MInstance memory d3m,
        D3MCommonConfig memory cfg,
        address gem
    ) private {
        bytes32 ilk = cfg.ilk;
        D3MHubLike hub = D3MHubLike(cfg.hub);
        D3MOracleLike oracle = D3MOracleLike(d3m.oracle);

        // Sanity checks
        require(oracle.vat() == address(dss.vat), "Oracle vat mismatch");
        require(oracle.ilk() == ilk, "Oracle ilk mismatch");

        hub.file(ilk, "pool", d3m.pool);
        hub.file(ilk, "plan", d3m.plan);
        hub.file(ilk, "tau", cfg.tau);

        oracle.file("hub", address(hub));

        dss.spotter.file(ilk, "pip", address(oracle));
        dss.spotter.file(ilk, "mat", 10 ** 27);
        uint256 previousIlkLine;
        if (cfg.existingIlk) {
            (,,, previousIlkLine,) = dss.vat.ilks(ilk);
        } else {
            dss.vat.init(ilk);
            dss.jug.init(ilk);
        }
        dss.vat.file(ilk, "line", cfg.gap);
        dss.vat.file("Line", dss.vat.Line() + cfg.gap - previousIlkLine);
        DssAutoLineAbstract(dss.chainlog.getAddress("MCD_IAM_AUTO_LINE")).setIlk
            ↪ (
            ilk,
            cfg.maxLine,
            cfg.gap,
            cfg.ttl
        );
        dss.spotter.poke(ilk);

        IlkRegistryAbstract(dss.chainlog.getAddress("ILK_REGISTRY")).put(
            ilk,
            address(hub),
            address(gem),
            GemAbstract(gem).decimals(),
            4,
            address(oracle),
            address(0),
            GemAbstract(gem).name(),
            GemAbstract(gem).symbol()
        );

        string memory clPrefix = ScriptTools.ilkToChainlogFormat(ilk);
        dss.chainlog.setAddress(ScriptTools.stringToBytes32(string(abi.
            ↪ encodePacked(clPrefix, "_POOL"))), d3m.pool);
        dss.chainlog.setAddress(ScriptTools.stringToBytes32(string(abi.
            ↪ encodePacked(clPrefix, "_PLAN"))), d3m.plan);
        dss.chainlog.setAddress(ScriptTools.stringToBytes32(string(abi.
            ↪ encodePacked(clPrefix, "_ORACLE"))), d3m.oracle);
    }
```

21.4.3 initAave(dss, d3m, cfg, aaveCfg)

```
    function initAave(
        DssInstance memory dss,
        D3MInstance memory d3m,
        D3MCommonConfig memory cfg,
        D3MAaveConfig memory aaveCfg
    ) internal {
        AavePlanLike plan = AavePlanLike(d3m.plan);
        AavePoolLike pool = AavePoolLike(d3m.pool);
        ADaiLike adai = ADaiLike(aaveCfg.adai);

        _init(dss, d3m, cfg, address(adai));

        // Sanity checks
        require(pool.hub() == cfg.hub, "Pool hub mismatch");
        require(pool.ilk() == cfg.ilk, "Pool ilk mismatch");
        require(pool.vat() == address(dss.vat), "Pool vat mismatch");
        require(pool.dai() == address(dss.dai), "Pool dai mismatch");
        require(pool.adai() == address(adai), "Pool adai mismatch");
        require(pool.stableDebt() == aaveCfg.stableDebt, "Pool stableDebt
            ↪ mismatch");
        require(pool.variableDebt() == aaveCfg.variableDebt, "Pool variableDebt
            ↪ mismatch");

        require(plan.adai() == address(adai), "Plan adai mismatch");
        require(plan.stableDebt() == aaveCfg.stableDebt, "Plan stableDebt
            ↪ mismatch");
        require(plan.variableDebt() == aaveCfg.variableDebt, "Plan variableDebt
            ↪ mismatch");
        require(plan.tack() == aaveCfg.tack, "Plan tack mismatch");
        require(plan.adaiRevision() == aaveCfg.adaiRevision, "Plan adaiRevision
            ↪ mismatch");
        require(adai.ATOKEN_REVISION() == aaveCfg.adaiRevision, "ADai
            ↪ adaiRevision mismatch");

        plan.rely(cfg.mom);
        pool.file("king", aaveCfg.king);
        plan.file("bar", aaveCfg.bar);
    }
```

21.4.4 initCompound(dss, d3m, cfg, compoundCfg)

```
    function initCompound(
        DssInstance memory dss,
        D3MInstance memory d3m,
        D3MCommonConfig memory cfg,
        D3MCompoundConfig memory compoundCfg
    ) internal {
        CompoundPlanLike plan = CompoundPlanLike(d3m.plan);
        CompoundPoolLike pool = CompoundPoolLike(d3m.pool);
        CDaiLike cdai = CDaiLike(compoundCfg.cdai);

        _init(dss, d3m, cfg, address(cdai));

        // Sanity checks
        require(pool.hub() == cfg.hub, "Pool hub mismatch");
        require(pool.ilk() == cfg.ilk, "Pool ilk mismatch");
        require(pool.vat() == address(dss.vat), "Pool vat mismatch");
        require(pool.dai() == address(dss.dai), "Pool dai mismatch");
        require(pool.comptroller() == compoundCfg.comptroller, "Pool comptroller
            ↪  mismatch");
        require(pool.comp() == compoundCfg.comp, "Pool comp mismatch");
        require(pool.cDai() == address(cdai), "Pool cDai mismatch");

        require(plan.tack() == compoundCfg.tack, "Plan tack mismatch");
        require(cdai.interestRateModel() == compoundCfg.tack, "CDai tack
            ↪ mismatch");
        require(plan.delegate() == compoundCfg.delegate, "Plan delegate mismatch
            ↪ ");
        require(cdai.implementation() == compoundCfg.delegate, "CDai delegate
            ↪ mismatch");
        require(plan.cDai() == address(cdai), "Plan cDai mismatch");

        plan.rely(cfg.mom);
        pool.file("king", compoundCfg.king);
        plan.file("barb", compoundCfg.barb);
    }
```

21.5 library DssExecLib

```
library DssExecLib {

    /*****************/
    /*** Constants ***/
    /*****************/
    address constant public LOG = 0xdA0Ab1e0017DEbCd72Be8599041a2aa3bA7e740F;

    uint256 constant internal WAD      = 10 ** 18;
    uint256 constant internal RAY      = 10 ** 27;
    uint256 constant internal RAD      = 10 ** 45;
    uint256 constant internal THOUSAND = 10 ** 3;
    uint256 constant internal MILLION  = 10 ** 6;

    uint256 constant internal BPS_ONE_PCT             = 100;
    uint256 constant internal BPS_ONE_HUNDRED_PCT     = 100 * BPS_ONE_PCT;
    uint256 constant internal RATES_ONE_HUNDRED_PCT   =
        ↪ 1000000021979553151239153027;

    /**********************/
    /*** Math Functions ***/

    /***************************/
    /*** Core Address Helpers ***/

    /***************************/
    /*** Changelog Management ***/
    /***************************/


    /**********************/
    /*** Authorizations ***/
    /**********************/

    /*****************************/
    /*** OfficeHours Management ***/
    /*****************************/

    /*************************/
    /*** Accumulating Rates ***/
    /*************************/

    /*********************/
    /*** Price Updates ***/
    /*********************/

    /***************************/
    /*** System Configuration ***/
    /***************************/

    /*****************************/
    /*** System Risk Parameters ***/
    /*****************************/
    // function setGlobalDebtCeiling(uint256 _amount) public {
        ↪ setGlobalDebtCeiling(vat(), _amount); }

    /****************************/
    /*** Collateral Management ***/
    /****************************/


    /************************/
    /*** Abacus Management ***/
    /************************/

    /************************/
    /*** Oracle Management ***/
    /*************************/

    /*****************************/
    /*** Direct Deposit Module ***/
    /*****************************/

    /*****************************/
    /*** Collateral Onboarding ***/
    /*****************************/

    /***************/
    /*** Payment ***/
    /***************/

    /************/
    /*** Misc ***/
    /************/
}
```

21.5.1 wdiv(x, y)

```
    /*********************/
    function wdiv(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = (x * WAD + y / 2) / y;
    }
```

21.5.2 rdiv(x, y)

```
    function rdiv(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = (x * RAY + y / 2) / y;
    }
```

21.5.3 dai()

```
    /***************************/
    function dai()        public view returns (address) { return
       ↪ getChangelogAddress("MCD_DAI"); }
```

21.5.4 mkr()

```
    function mkr()        public view returns (address) { return
       ↪ getChangelogAddress("MCD_GOV"); }
```

21.5.5 vat()

```
    function vat()        public view returns (address) { return
       ↪ getChangelogAddress("MCD_VAT"); }
```

21.5.6 cat()

```
    function cat()        public view returns (address) { return
       ↪ getChangelogAddress("MCD_CAT"); }
```

21.5.7 dog()

```
    function dog()        public view returns (address) { return
       ↪ getChangelogAddress("MCD_DOG"); }
```

21.5.8 jug()

```
function jug()          public view returns (address) { return
    ↪ getChangelogAddress("MCD_JUG"); }
```

21.5.9 pot()

```
function pot()          public view returns (address) { return
    ↪ getChangelogAddress("MCD_POT"); }
```

21.5.10 vow()

```
function vow()          public view returns (address) { return
    ↪ getChangelogAddress("MCD_VOW"); }
```

21.5.11 end()

```
function end()          public view returns (address) { return
    ↪ getChangelogAddress("MCD_END"); }
```

21.5.12 esm()

```
function esm()          public view returns (address) { return
    ↪ getChangelogAddress("MCD_ESM"); }
```

21.5.13 reg()

```
function reg()          public view returns (address) { return
    ↪ getChangelogAddress("ILK_REGISTRY"); }
```

21.5.14 spotter()

```
function spotter()      public view returns (address) { return
    ↪ getChangelogAddress("MCD_SPOT"); }
```

21.5.15 flap()

```
function flap()         public view returns (address) { return
    ↪ getChangelogAddress("MCD_FLAP"); }
```

21.5.16 flop()

```
function flop()         public view returns (address) { return
    ↪ getChangelogAddress("MCD_FLOP"); }
```

21.5.17 osmMom()

```
function osmMom()       public view returns (address) { return
    ↪ getChangelogAddress("OSM_MOM"); }
```

21.5.18 govGuard()

```
function govGuard()   public view returns (address) { return
    ↪ getChangelogAddress("GOV_GUARD"); }
```

21.5.19 flipperMom()

```
function flipperMom() public view returns (address) { return
    ↪ getChangelogAddress("FLIPPER_MOM"); }
```

21.5.20 clipperMom()

```
function clipperMom() public view returns (address) { return
    ↪ getChangelogAddress("CLIPPER_MOM"); }
```

21.5.21 pauseProxy()

```
function pauseProxy() public view returns (address) { return
    ↪ getChangelogAddress("MCD_PAUSE_PROXY"); }
```

21.5.22 autoLine()

```
function autoLine()   public view returns (address) { return
    ↪ getChangelogAddress("MCD_IAM_AUTO_LINE"); }
```

21.5.23 daiJoin()

```
function daiJoin()    public view returns (address) { return
    ↪ getChangelogAddress("MCD_JOIN_DAI"); }
```

21.5.24 lerpFab()

```
function lerpFab()    public view returns (address) { return
    ↪ getChangelogAddress("LERP_FAB"); }
```

21.5.25 clip(_ilk)

```
function clip(bytes32 _ilk) public view returns (address _clip) {
    _clip = RegistryLike(reg()).xlip(_ilk);
}
```

21.5.26 flip(_ilk)

```
function flip(bytes32 _ilk) public view returns (address _flip) {
    _flip = RegistryLike(reg()).xlip(_ilk);
}
```

21.5.27 calc(_ilk)

```
function calc(bytes32 _ilk) public view returns (address _calc) {
    _calc = ClipLike(clip(_ilk)).calc();
}
```

21.5.28 getChangelogAddress(_key)

```
function getChangelogAddress(bytes32 _key) public view returns (address) {
    return ChainlogLike(LOG).getAddress(_key);
}
```

21.5.29 setChangelogAddress(_key, _val) X

```
/**
    @dev Set an address in the MCD on-chain changelog.
    @param _key Access key for the address (e.g. "MCD_VAT")
    @param _val The address associated with the _key
*/
function setChangelogAddress(bytes32 _key, address _val) public {
    ChainlogLike(LOG).setAddress(_key, _val);
}
```

21.5.30 setChangelogVersion(_version) X

```
/**
    @dev Set version in the MCD on-chain changelog.
    @param _version Changelog version (e.g. "1.1.2")
*/
function setChangelogVersion(string memory _version) public {
    ChainlogLike(LOG).setVersion(_version);
}
```

21.5.31 setChangelogIPFS(_ipfsHash) X

```
/**
    @dev Set IPFS hash of IPFS changelog in MCD on-chain changelog.
    @param _ipfsHash IPFS hash (e.g. "
        ↪ QmefQMseb3AiTapiAKKexdKHig8wroKuZbmLtPLv4u2YwW")
*/
function setChangelogIPFS(string memory _ipfsHash) public {
    ChainlogLike(LOG).setIPFS(_ipfsHash);
}
```

21.5.32 setChangelogSHA256(_SHA256Sum) X

```
/**
    @dev Set SHA256 hash in MCD on-chain changelog.
    @param _SHA256Sum SHA256 hash (e.g. "
        ↪ e42dc9d043a57705f3f097099e6b2de4230bca9a020c797508da079f9079e35b
        ↪ ")
*/
function setChangelogSHA256(string memory _SHA256Sum) public {
    ChainlogLike(LOG).setSha256sum(_SHA256Sum);
}
```

21.5.33 authorize(_base, _ward) X

```
/**
    @dev Give an address authorization to perform auth actions on the
        ↪ contract.
    @param _base   The address of the contract where the authorization will
        ↪ be set
    @param _ward   Address to be authorized
*/
function authorize(address _base, address _ward) public {
    Authorizable(_base).rely(_ward);
}
```

21.5.34 deauthorize(_base, _ward) X

```
    /**
        @dev Revoke contract authorization from an address.
        @param _base   The address of the contract where the authorization will
           ↪ be revoked
        @param _ward   Address to be deauthorized
    */
    function deauthorize(address _base, address _ward) public {
        Authorizable(_base).deny(_ward);
    }
```

21.5.35 setAuthority(_base, _authority) X

```
    /**
        @dev Give an address authorization to perform auth actions on the
           ↪ contract.
        @param _base   The address of the contract with a `setAuthority` pattern
        @param _authority   Address to be authorized
    */
    function setAuthority(address _base, address _authority) public {
        Authorizable(_base).setAuthority(_authority);
    }
```

21.5.36 delegateVat(_usr) X

```
    /**
        @dev Delegate vat authority to the specified address.
        @param _usr Address to be authorized
    */
    function delegateVat(address _usr) public {
        DssVat(vat()).hope(_usr);
    }
```

21.5.37 undelegateVat(_usr) X

```
    /**
        @dev Revoke vat authority to the specified address.
        @param _usr Address to be deauthorized
    */
    function undelegateVat(address _usr) public {
        DssVat(vat()).nope(_usr);
    }
```

21.5.38 canCast(_ts, _officeHours)

```
    /**
        @dev Returns true if a time is within office hours range
        @param _ts           The timestamp to check, usually block.timestamp
        @param _officeHours  true if office hours is enabled.
        @return              true if time is in castable range
    */
    function canCast(uint40 _ts, bool _officeHours) public pure returns (bool) {
        if (_officeHours) {
            uint256 day = (_ts / 1 days + 3) % 7;
            if (day >= 5)                 { return false; }  // Can only be cast
               ↪ on a weekday
            uint256 hour = _ts / 1 hours % 24;
            if (hour < 14 || hour >= 21)  { return false; }  // Outside office
               ↪ hours
        }
        return true;
    }
```

21.5.39 nextCastTime(_eta, _ts, _officeHours)

```
/**
    @dev Calculate the next available cast time in epoch seconds
    @param _eta          The scheduled time of the spell plus the pause
        ↪ delay
    @param _ts           The current timestamp, usually block.timestamp
    @param _officeHours  true if office hours is enabled.
    @return castTime     The next available cast timestamp
*/
function nextCastTime(uint40 _eta, uint40 _ts, bool _officeHours) public
    ↪ pure returns (uint256 castTime) {
    require(_eta != 0);  // "DssExecLib/invalid eta"
    require(_ts  != 0);  // "DssExecLib/invalid ts"
    castTime = _ts > _eta ? _ts : _eta; // Any day at XX:YY

    if (_officeHours) {
        uint256 day    = (castTime / 1 days + 3) % 7;
        uint256 hour   = castTime / 1 hours % 24;
        uint256 minute = castTime / 1 minutes % 60;
        uint256 second = castTime % 60;

        if (day >= 5) {
            castTime += (6 - day) * 1 days;                 // Go to Sunday
                ↪ XX:YY
            castTime += (24 - hour + 14) * 1 hours;         // Go to 14:YY
                ↪ UTC Monday
            castTime -= minute * 1 minutes + second;        // Go to 14:00
                ↪ UTC
        } else {
            if (hour >= 21) {
                if (day == 4) castTime += 2 days;           // If Friday,
                    ↪ fast forward to Sunday XX:YY
                castTime += (24 - hour + 14) * 1 hours;     // Go to 14:YY
                    ↪ UTC next day
                castTime -= minute * 1 minutes + second;    // Go to 14:00
                    ↪ UTC
            } else if (hour < 14) {
                castTime += (14 - hour) * 1 hours;          // Go to 14:YY
                    ↪ UTC same day
                castTime -= minute * 1 minutes + second;    // Go to 14:00
                    ↪ UTC
            }
        }
    }
}
```

21.5.40 accumulateDSR() X

```
/**
    @dev Update rate accumulation for the Dai Savings Rate (DSR).
*/
function accumulateDSR() public {
    Drippable(pot()).drip();
}
```

21.5.41 accumulateCollateralStabilityFees(_ilk) X

```
/**
    @dev Update rate accumulation for the stability fees of a given
        ↪ collateral type.
    @param _ilk   Collateral type
*/
function accumulateCollateralStabilityFees(bytes32 _ilk) public {
    Drippable(jug()).drip(_ilk);
}
```

21.5.42 updateCollateralPrice(_ilk) X

```
/**
    @dev Update price of a given collateral type.
    @param _ilk   Collateral type
*/
function updateCollateralPrice(bytes32 _ilk) public {
    Pricing(spotter()).poke(_ilk);
}
```

21.5.43 setContract(_base, _what, _addr) X

```
/**
    @dev Set a contract in another contract, defining the relationship (ex.
       ↪ set a new Calc contract in Clip)
    @param _base   The address of the contract where the new contract
       ↪ address will be filed
    @param _what   Name of contract to file
    @param _addr   Address of contract to file
*/
function setContract(address _base, bytes32 _what, address _addr) public {
    Fileable(_base).file(_what, _addr);
}
```

21.5.44 setContract(_base, _ilk, _what, _addr) X

```
/**
    @dev Set a contract in another contract, defining the relationship (ex.
       ↪ set a new Calc contract in a Clip)
    @param _base   The address of the contract where the new contract
       ↪ address will be filed
    @param _ilk    Collateral type
    @param _what   Name of contract to file
    @param _addr   Address of contract to file
*/
function setContract(address _base, bytes32 _ilk, bytes32 _what, address
   ↪ _addr) public {
    Fileable(_base).file(_ilk, _what, _addr);
}
```

21.5.45 setValue(_base, _what, _amt) X

```
/**
    @dev Set a value in a contract, via a governance authorized File pattern
       ↪ .
    @param _base   The address of the contract where the new contract
       ↪ address will be filed
    @param _what   Name of tag for the value (e.x. "Line")
    @param _amt    The value to set or update
*/
function setValue(address _base, bytes32 _what, uint256 _amt) public {
    Fileable(_base).file(_what, _amt);
}
```

21.5.46 setValue(_base, _ilk, _what, _amt) X

```
/**
    @dev Set an ilk-specific value in a contract, via a governance
       ↪ authorized File pattern.
    @param _base   The address of the contract where the new value will be
       ↪ filed
    @param _ilk    Collateral type
```

```
        @param _what   Name of tag for the value (e.x. "Line")
        @param _amt    The value to set or update
    */
    function setValue(address _base, bytes32 _ilk, bytes32 _what, uint256 _amt)
        ↪ public {
        Fileable(_base).file(_ilk, _what, _amt);
    }
```

21.5.47 setGlobalDebtCeiling(_amount) X

```
    /**
        @dev Set the global debt ceiling. Amount will be converted to the
            ↪ correct internal precision.
        @param _amount The amount to set in DAI (ex. 10m DAI amount == 10000000)
    */
    function setGlobalDebtCeiling(uint256 _amount) public {
        require(_amount < WAD);  // "LibDssExec/incorrect-global-Line-precision"
        setValue(vat(), "Line", _amount * RAD);
    }
```

21.5.48 increaseGlobalDebtCeiling(_amount) X

```
    /**
        @dev Increase the global debt ceiling by a specific amount. Amount will
            ↪ be converted to the correct internal precision.
        @param _amount The amount to add in DAI (ex. 10m DAI amount == 10000000)
    */
    function increaseGlobalDebtCeiling(uint256 _amount) public {
        require(_amount < WAD);  // "LibDssExec/incorrect-Line-increase-
            ↪ precision"
        address _vat = vat();
        setValue(_vat, "Line", DssVat(_vat).Line() + _amount * RAD);
    }
```

21.5.49 decreaseGlobalDebtCeiling(_amount) X

```
    /**
        @dev Decrease the global debt ceiling by a specific amount. Amount will
            ↪ be converted to the correct internal precision.
        @param _amount The amount to reduce in DAI (ex. 10m DAI amount ==
            ↪ 10000000)
    */
    function decreaseGlobalDebtCeiling(uint256 _amount) public {
        require(_amount < WAD);  // "LibDssExec/incorrect-Line-decrease-
            ↪ precision"
        address _vat = vat();
        setValue(_vat, "Line", DssVat(_vat).Line() - _amount * RAD);
    }
```

21.5.50 setDSR(_rate, _doDrip) X

```
    /**
        @dev Set the Dai Savings Rate. See: docs/rates.txt
        @param _rate   The accumulated rate (ex. 4% =>
            ↪ 1000000001243680656318820312)
        @param _doDrip `true` to accumulate interest owed
    */
    function setDSR(uint256 _rate, bool _doDrip) public {
        require((_rate >= RAY) && (_rate <= RATES_ONE_HUNDRED_PCT));  // "
            ↪ LibDssExec/dsr-out-of-bounds"
        if (_doDrip) Drippable(pot()).drip();
        setValue(pot(), "dsr", _rate);
    }
```

21.5.51 setSurplusAuctionAmount(_amount) X

```
    /**
        @dev Set the DAI amount for system surplus auctions. Amount will be
            ↪ converted to the correct internal precision.
        @param _amount The amount to set in DAI (ex. 10m DAI amount == 10000000)
    */
    function setSurplusAuctionAmount(uint256 _amount) public {
        require(_amount < WAD);  // "LibDssExec/incorrect-vow-bump-precision"
        setValue(vow(), "bump", _amount * RAD);
    }
```

21.5.52 setSurplusBuffer(_amount) X

```
    /**
        @dev Set the DAI amount for system surplus buffer, must be exceeded
            ↪ before surplus auctions start. Amount will be converted to the
            ↪ correct internal precision.
        @param _amount The amount to set in DAI (ex. 10m DAI amount == 10000000)
    */
    function setSurplusBuffer(uint256 _amount) public {
        require(_amount < WAD);  // "LibDssExec/incorrect-vow-hump-precision"
        setValue(vow(), "hump", _amount * RAD);
    }
```

21.5.53 setMinSurplusAuctionBidIncrease(_pct_bps) X

```
    /**
        @dev Set minimum bid increase for surplus auctions. Amount will be
            ↪ converted to the correct internal precision.
        @dev Equation used for conversion is (1 + pct / 10,000) * WAD
        @param _pct_bps The pct, in basis points, to set in integer form (x100).
            ↪  (ex. 5% = 5 * 100 = 500)
    */
    function setMinSurplusAuctionBidIncrease(uint256 _pct_bps) public {
        require(_pct_bps < BPS_ONE_HUNDRED_PCT);  // "LibDssExec/incorrect-flap-
            ↪ beg-precision"
        setValue(flap(), "beg", WAD + wdiv(_pct_bps, BPS_ONE_HUNDRED_PCT));
    }
```

21.5.54 setSurplusAuctionBidDuration(_duration) X

```
    /**
        @dev Set bid duration for surplus auctions.
        @param _duration Amount of time for bids. (in seconds)
    */
    function setSurplusAuctionBidDuration(uint256 _duration) public {
        setValue(flap(), "ttl", _duration);
    }
```

21.5.55 setSurplusAuctionDuration(_duration) X

```
    /**
        @dev Set total auction duration for surplus auctions.
        @param _duration Amount of time for auctions. (in seconds)
    */
    function setSurplusAuctionDuration(uint256 _duration) public {
        setValue(flap(), "tau", _duration);
    }
```

21.5.56 setDebtAuctionDelay(_duration) X

```
/**
    @dev Set the number of seconds that pass before system debt is auctioned
        ↪ for MKR tokens.
    @param _duration Duration in seconds
*/
function setDebtAuctionDelay(uint256 _duration) public {
    setValue(vow(), "wait", _duration);
}
```

21.5.57 setDebtAuctionDAIAmount(_amount) X

```
/**
    @dev Set the DAI amount for system debt to be covered by each debt
        ↪ auction. Amount will be converted to the correct internal
        ↪ precision.
    @param _amount The amount to set in DAI (ex. 10m DAI amount == 10000000)
*/
function setDebtAuctionDAIAmount(uint256 _amount) public {
    require(_amount < WAD);  // "LibDssExec/incorrect-vow-sump-precision"
    setValue(vow(), "sump", _amount * RAD);
}
```

21.5.58 setDebtAuctionMKRAmount(_amount) X

```
/**
    @dev Set the starting MKR amount to be auctioned off to cover system
        ↪ debt in debt auctions. Amount will be converted to the correct
        ↪ internal precision.
    @param _amount The amount to set in MKR (ex. 250 MKR amount == 250)
*/
function setDebtAuctionMKRAmount(uint256 _amount) public {
    require(_amount < WAD);  // "LibDssExec/incorrect-vow-dump-precision"
    setValue(vow(), "dump", _amount * WAD);
}
```

21.5.59 setMinDebtAuctionBidIncrease(_pct_bps) X

```
/**
    @dev Set minimum bid increase for debt auctions. Amount will be
        ↪ converted to the correct internal precision.
    @dev Equation used for conversion is (1 + pct / 10,000) * WAD
    @param _pct_bps    The pct, in basis points, to set in integer form (
        ↪ x100). (ex. 5% = 5 * 100 = 500)
*/
function setMinDebtAuctionBidIncrease(uint256 _pct_bps) public {
    require(_pct_bps < BPS_ONE_HUNDRED_PCT);  // "LibDssExec/incorrect-flop-
        ↪ beg-precision"
    setValue(flop(), "beg", WAD + wdiv(_pct_bps, BPS_ONE_HUNDRED_PCT));
}
```

21.5.60 setDebtAuctionBidDuration(_duration) X

```
/**
    @dev Set bid duration for debt auctions.
    @param _duration Amount of time for bids. (seconds)
*/
function setDebtAuctionBidDuration(uint256 _duration) public {
    require(_duration < type(uint48).max);  // "LibDssExec/incorrect-flop-
        ↪ ttl-precision"
    setValue(flop(), "ttl", _duration);
}
```

21.5.61 setDebtAuctionDuration(_duration) X

```
    /**
        @dev Set total auction duration for debt auctions.
        @param _duration Amount of time for auctions. (seconds)
    */
    function setDebtAuctionDuration(uint256 _duration) public {
        require(_duration < type(uint48).max);  // "LibDssExec/incorrect-flop-
            ↪ tau-precision"
        setValue(flop(), "tau", _duration);
    }
```

21.5.62 setDebtAuctionMKRIncreaseRate(_pct_bps) X

```
    /**
        @dev Set the rate of increasing amount of MKR out for auction during
            ↪ debt auctions. Amount will be converted to the correct internal
            ↪ precision.
        @dev MKR amount is increased by this rate every "tick" (if auction
            ↪ duration has passed and no one has bid on the MKR)
        @dev Equation used for conversion is (1 + pct / 10,000) * WAD
        @param _pct_bps    The pct, in basis points, to set in integer form (
            ↪ x100). (ex. 5% = 5 * 100 = 500)
    */
    function setDebtAuctionMKRIncreaseRate(uint256 _pct_bps) public {
        require(_pct_bps < BPS_ONE_HUNDRED_PCT);  // "LibDssExec/incorrect-flop-
            ↪ pad-precision"
        setValue(flop(), "pad", WAD + wdiv(_pct_bps, BPS_ONE_HUNDRED_PCT));
    }
```

21.5.63 setMaxTotalDAILiquidationAmount(_amount) X

```
    /**
        @dev Set the maximum total DAI amount that can be out for liquidation in
            ↪  the system at any point. Amount will be converted to the correct
            ↪  internal precision.
        @param _amount The amount to set in DAI (ex. 250,000 DAI amount ==
            ↪ 250000)
    */
    function setMaxTotalDAILiquidationAmount(uint256 _amount) public {
        require(_amount < WAD);  // "LibDssExec/incorrect-dog-Hole-precision"
        setValue(dog(), "Hole", _amount * RAD);
    }
```

21.5.64 setMaxTotalDAILiquidationAmountLEGACY(_amount) X

```
    /**
        @dev (LIQ 1.2) Set the maximum total DAI amount that can be out for
            ↪ liquidation in the system at any point. Amount will be converted
            ↪ to the correct internal precision.
        @param _amount The amount to set in DAI (ex. 250,000 DAI amount ==
            ↪ 250000)
    */
    function setMaxTotalDAILiquidationAmountLEGACY(uint256 _amount) public {
        require(_amount < WAD);  // "LibDssExec/incorrect-cat-box-amount"
        setValue(cat(), "box", _amount * RAD);
    }
```

21.5.65 setEmergencyShutdownProcessingTime(_duration) X

```
    /**
        @dev Set the duration of time that has to pass during emergency shutdown
            ↪ before collateral can start being claimed by DAI holders.
        @param _duration Time in seconds to set for ES processing time
    */
    function setEmergencyShutdownProcessingTime(uint256 _duration) public {
        setValue(end(), "wait", _duration);
    }
```

21.5.66 setGlobalStabilityFee(_rate) X

```
    /**
        @dev Set the global stability fee (is not typically used, currently is
            ↪ 0).
            Many of the settings that change weekly rely on the rate accumulator
            described at https://docs.makerdao.com/smart-contract-modules/rates-
                ↪ module
            To check this yourself, use the following rate calculation (example
                ↪ 8%):

            $ bc -l <<< 'scale=27; e( l(1.08)/(60 * 60 * 24 * 365) )'

            A table of rates can also be found at:
            https://ipfs.io/ipfs/QmefQMseb3AiTapiAKKexdKHig8wroKuZbmLtPLv4u2YwW
        @param _rate   The accumulated rate (ex. 4% =>
            ↪ 1000000001243680656318820312)
    */
    function setGlobalStabilityFee(uint256 _rate) public {
        require((_rate >= RAY) && (_rate <= RATES_ONE_HUNDRED_PCT));  // "
            ↪ LibDssExec/global-stability-fee-out-of-bounds"
        setValue(jug(), "base", _rate);
    }
```

21.5.67 setDAIReferenceValue(_value) X

```
    /**
        @dev Set the value of DAI in the reference asset (e.g. $1 per DAI).
            ↪ Value will be converted to the correct internal precision.
        @dev Equation used for conversion is value * RAY / 1000
        @param _value The value to set as integer (x1000) (ex. $1.025 == 1025)
    */
    function setDAIReferenceValue(uint256 _value) public {
        require(_value < WAD);  // "LibDssExec/incorrect-par-precision"
        setValue(spotter(), "par", rdiv(_value, 1000));
    }
```

21.5.68 setIlkDebtCeiling(_ilk, _amount) X

```
    /**
        @dev Set a collateral debt ceiling. Amount will be converted to the
            ↪ correct internal precision.
        @param _ilk    The ilk to update (ex. bytes32("ETH-A"))
        @param _amount The amount to set in DAI (ex. 10m DAI amount == 10000000)
    */
    function setIlkDebtCeiling(bytes32 _ilk, uint256 _amount) public {
        require(_amount < WAD);  // "LibDssExec/incorrect-ilk-line-precision"
        setValue(vat(), _ilk, "line", _amount * RAD);
    }
```

21.5.69 increaseIlkDebtCeiling(_ilk, _amount, _global) X

```
    /**
        @dev Increase a collateral debt ceiling. Amount will be converted to the
            ↪ correct internal precision.
        @param _ilk    The ilk to update (ex. bytes32("ETH-A"))
        @param _amount The amount to increase in DAI (ex. 10m DAI amount ==
            ↪ 10000000)
        @param _global If true, increases the global debt ceiling by _amount
    */
    function increaseIlkDebtCeiling(bytes32 _ilk, uint256 _amount, bool _global)
        ↪ public {
        require(_amount < WAD);  // "LibDssExec/incorrect-ilk-line-precision"
        address _vat = vat();
        (,,,uint256 line_,) = DssVat(_vat).ilks(_ilk);
        setValue(_vat, _ilk, "line", line_ + _amount * RAD);
        if (_global) { increaseGlobalDebtCeiling(_amount); }
    }
```

21.5.70 decreaseIlkDebtCeiling(_ilk, _amount, _global) X

```
    /**
        @dev Decrease a collateral debt ceiling. Amount will be converted to the
            ↪ correct internal precision.
        @param _ilk    The ilk to update (ex. bytes32("ETH-A"))
        @param _amount The amount to decrease in DAI (ex. 10m DAI amount ==
            ↪ 10000000)
        @param _global If true, decreases the global debt ceiling by _amount
    */
    function decreaseIlkDebtCeiling(bytes32 _ilk, uint256 _amount, bool _global)
        ↪ public {
        require(_amount < WAD);  // "LibDssExec/incorrect-ilk-line-precision"
        address _vat = vat();
        (,,,uint256 line_,) = DssVat(_vat).ilks(_ilk);
        setValue(_vat, _ilk, "line", line_ - _amount * RAD);
        if (_global) { decreaseGlobalDebtCeiling(_amount); }
    }
```

21.5.71 setRWAIlkDebtCeiling(_ilk, _ceiling, _price) X

```
    /**
        @dev Set a RWA collateral debt ceiling by specifying its new oracle
            ↪ price.
        @param _ilk      The ilk to update (ex. bytes32("ETH-A"))
        @param _ceiling  The new debt ceiling in natural units (e.g. set 10m DAI
            ↪  as 10_000_000)
        @param _price    The new oracle price in natural units
        @dev note: _price should enable DAI to be drawn over the loan period
            ↪ while taking into
                 account the configured ink amount, interest rate and
                     ↪ liquidation ratio
        @dev note: _price * WAD should be greater than or equal to the current
            ↪ oracle price
    */
    function setRWAIlkDebtCeiling(bytes32 _ilk, uint256 _ceiling, uint256 _price
        ↪ ) public {
        require(_price < WAD);
        setIlkDebtCeiling(_ilk, _ceiling);
        RwaOracleLike(getChangelogAddress("MIP21_LIQUIDATION_ORACLE")).bump(_ilk
            ↪ , _price * WAD);
        updateCollateralPrice(_ilk);
    }
```

21.5.72 setIlkAutoLineParameters(_ilk, _amount, _gap, _ttl) X

```
    /**
        @dev Set the parameters for an ilk in the "MCD_IAM_AUTO_LINE" auto-line
        @param _ilk    The ilk to update (ex. bytes32("ETH-A"))
        @param _amount The Maximum value (ex. 100m DAI amount == 100000000)
        @param _gap    The amount of Dai per step (ex. 5m Dai == 5000000)
        @param _ttl    The amount of time (in seconds)
    */
    function setIlkAutoLineParameters(bytes32 _ilk, uint256 _amount, uint256
       ↪ _gap, uint256 _ttl) public {
        require(_amount < WAD);  // "LibDssExec/incorrect-auto-line-amount-
           ↪ precision"
        require(_gap < WAD);  // "LibDssExec/incorrect-auto-line-gap-precision"
        IAMLike(autoLine()).setIlk(_ilk, _amount * RAD, _gap * RAD, _ttl);
    }
```

21.5.73 setIlkAutoLineDebtCeiling(_ilk, _amount) X

```
    /**
        @dev Set the debt ceiling for an ilk in the "MCD_IAM_AUTO_LINE" auto-
           ↪ line without updating the time values
        @param _ilk    The ilk to update (ex. bytes32("ETH-A"))
        @param _amount The Maximum value (ex. 100m DAI amount == 100000000)
    */
    function setIlkAutoLineDebtCeiling(bytes32 _ilk, uint256 _amount) public {
        address _autoLine = autoLine();
        (, uint256 gap, uint48 ttl,,) = IAMLike(_autoLine).ilks(_ilk);
        require(gap != 0 && ttl != 0);  // "LibDssExec/auto-line-not-configured"
        IAMLike(_autoLine).setIlk(_ilk, _amount * RAD, uint256(gap), uint256(ttl
           ↪ ));
    }
```

21.5.74 removeIlkFromAutoLine(_ilk) X

```
    /**
        @dev Remove an ilk in the "MCD_IAM_AUTO_LINE" auto-line
        @param _ilk    The ilk to remove (ex. bytes32("ETH-A"))
    */
    function removeIlkFromAutoLine(bytes32 _ilk) public {
        IAMLike(autoLine()).remIlk(_ilk);
    }
```

21.5.75 setIlkMinVaultAmount(_ilk, _amount) X

```
    /**
        @dev Set a collateral minimum vault amount. Amount will be converted to
           ↪ the correct internal precision.
        @param _ilk    The ilk to update (ex. bytes32("ETH-A"))
        @param _amount The amount to set in DAI (ex. 10m DAI amount == 10000000)
    */
    function setIlkMinVaultAmount(bytes32 _ilk, uint256 _amount) public {
        require(_amount < WAD);  // "LibDssExec/incorrect-ilk-dust-precision"
        (,, uint256 _hole,) = DogLike(dog()).ilks(_ilk);
        require(_amount <= _hole / RAD);  // Ensure ilk.hole >= dust
        setValue(vat(), _ilk, "dust", _amount * RAD);
        (bool ok,) = clip(_ilk).call(abi.encodeWithSignature("upchost()")); ok;
    }
```

21.5.76 setIlkLiquidationPenalty(_ilk, _pct_bps) X

```
    /**
        @dev Set a collateral liquidation penalty. Amount will be converted to
           ↪ the correct internal precision.
        @dev Equation used for conversion is (1 + pct / 10,000) * WAD
        @param _ilk    The ilk to update (ex. bytes32("ETH-A"))
        @param _pct_bps    The pct, in basis points, to set in integer form (
            ↪ x100). (ex. 10.25% = 10.25 * 100 = 1025)
    */
    function setIlkLiquidationPenalty(bytes32 _ilk, uint256 _pct_bps) public {
        require(_pct_bps < BPS_ONE_HUNDRED_PCT);  // "LibDssExec/incorrect-ilk-
            ↪ chop-precision"
        setValue(dog(), _ilk, "chop", WAD + wdiv(_pct_bps, BPS_ONE_HUNDRED_PCT))
            ↪ ;
        (bool ok,) = clip(_ilk).call(abi.encodeWithSignature("upchost()")); ok;
    }
```

21.5.77 setIlkMaxLiquidationAmount(_ilk, _amount) X

```
    /**
        @dev Set max DAI amount for liquidation per vault for collateral. Amount
            ↪ will be converted to the correct internal precision.
        @param _ilk    The ilk to update (ex. bytes32("ETH-A"))
        @param _amount The amount to set in DAI (ex. 10m DAI amount == 10000000)
    */
    function setIlkMaxLiquidationAmount(bytes32 _ilk, uint256 _amount) public {
        require(_amount < WAD);  // "LibDssExec/incorrect-ilk-hole-precision"
        setValue(dog(), _ilk, "hole", _amount * RAD);
    }
```

21.5.78 setIlkLiquidationRatio(_ilk, _pct_bps) X

```
    /**
        @dev Set a collateral liquidation ratio. Amount will be converted to the
            ↪ correct internal precision.
        @dev Equation used for conversion is pct * RAY / 10,000
        @param _ilk    The ilk to update (ex. bytes32("ETH-A"))
        @param _pct_bps    The pct, in basis points, to set in integer form (
            ↪ x100). (ex. 150% = 150 * 100 = 15000)
    */
    function setIlkLiquidationRatio(bytes32 _ilk, uint256 _pct_bps) public {
        require(_pct_bps < 10 * BPS_ONE_HUNDRED_PCT); // "LibDssExec/incorrect-
            ↪ ilk-mat-precision" // Fails if pct >= 1000%
        require(_pct_bps >= BPS_ONE_HUNDRED_PCT); // the liquidation ratio has
            ↪ to be bigger or equal to 100%
        setValue(spotter(), _ilk, "mat", rdiv(_pct_bps, BPS_ONE_HUNDRED_PCT));
    }
```

21.5.79 setStartingPriceMultiplicativeFactor(_ilk, _pct_bps) X

```
    /**
        @dev Set an auction starting multiplier. Amount will be converted to the
            ↪ correct internal precision.
        @dev Equation used for conversion is pct * RAY / 10,000
        @param _ilk      The ilk to update (ex. bytes32("ETH-A"))
        @param _pct_bps  The pct, in basis points, to set in integer form (x100)
            ↪ . (ex. 1.3x starting multiplier = 130% = 13000)
    */
    function setStartingPriceMultiplicativeFactor(bytes32 _ilk, uint256 _pct_bps
        ↪ ) public {
        require(_pct_bps < 10 * BPS_ONE_HUNDRED_PCT); // "LibDssExec/incorrect-
            ↪ ilk-mat-precision" // Fails if gt 10x
        require(_pct_bps >= BPS_ONE_HUNDRED_PCT); // fail if start price is less
            ↪ than OSM price
        setValue(clip(_ilk), "buf", rdiv(_pct_bps, BPS_ONE_HUNDRED_PCT));
    }
```

21.5.80 setAuctionTimeBeforeReset(_ilk, _duration) X

```
    /**
        @dev Set the amout of time before an auction resets.
        @param _ilk      The ilk to update (ex. bytes32("ETH-A"))
        @param _duration Amount of time before auction resets (in seconds).
    */
    function setAuctionTimeBeforeReset(bytes32 _ilk, uint256 _duration) public {
        setValue(clip(_ilk), "tail", _duration);
    }
```

21.5.81 setAuctionPermittedDrop(_ilk, _pct_bps) X

```
    /**
        @dev Percentage drop permitted before auction reset
        @param _ilk     The ilk to update (ex. bytes32("ETH-A"))
        @param _pct_bps The pct, in basis points, of drop to permit (x100).
    */
    function setAuctionPermittedDrop(bytes32 _ilk, uint256 _pct_bps) public {
        require(_pct_bps < BPS_ONE_HUNDRED_PCT); // "LibDssExec/incorrect-clip-
            ↪ cusp-value"
        setValue(clip(_ilk), "cusp", rdiv(_pct_bps, BPS_ONE_HUNDRED_PCT));
    }
```

21.5.82 setKeeperIncentivePercent(_ilk, _pct_bps) X

```
    /**
        @dev Percentage of tab to suck from vow to incentivize keepers. Amount
            ↪ will be converted to the correct internal precision.
        @param _ilk     The ilk to update (ex. bytes32("ETH-A"))
        @param _pct_bps The pct, in basis points, of the tab to suck. (0.01% ==
            ↪ 1)
    */
    function setKeeperIncentivePercent(bytes32 _ilk, uint256 _pct_bps) public {
        require(_pct_bps < BPS_ONE_HUNDRED_PCT); // "LibDssExec/incorrect-clip-
            ↪ chip-precision"
        setValue(clip(_ilk), "chip", wdiv(_pct_bps, BPS_ONE_HUNDRED_PCT));
    }
```

21.5.83 setKeeperIncentiveFlatRate(_ilk, _amount) X

```
    /**
        @dev Set max DAI amount for flat rate keeper incentive. Amount will be
            ↪ converted to the correct internal precision.
        @param _ilk    The ilk to update (ex. bytes32("ETH-A"))
        @param _amount The amount to set in DAI (ex. 1000 DAI amount == 1000)
    */
    function setKeeperIncentiveFlatRate(bytes32 _ilk, uint256 _amount) public {
        require(_amount < WAD); // "LibDssExec/incorrect-clip-tip-precision"
        setValue(clip(_ilk), "tip", _amount * RAD);
    }
```

21.5.84 setLiquidationBreakerPriceTolerance(_clip, _pct_bps) X

```
    /**
        @dev Sets the circuit breaker price tolerance in the clipper mom.
            This is somewhat counter-intuitive,
             to accept a 25% price drop, use a value of 75%
        @param _clip    The clipper to set the tolerance for
        @param _pct_bps The pct, in basis points, to set in integer form (x100).
            ↪ (ex. 5% = 5 * 100 = 500)
    */
    function setLiquidationBreakerPriceTolerance(address _clip, uint256 _pct_bps
       ↪ ) public {
        require(_pct_bps < BPS_ONE_HUNDRED_PCT);  // "LibDssExec/incorrect-
            ↪ clippermom-price-tolerance"
        MomLike(clipperMom()).setPriceTolerance(_clip, rdiv(_pct_bps,
            ↪ BPS_ONE_HUNDRED_PCT));
    }
```

21.5.85 setIlkStabilityFee(_ilk, _rate, _doDrip) X

```
    /**
        @dev Set the stability fee for a given ilk.
            Many of the settings that change weekly rely on the rate accumulator
            described at https://docs.makerdao.com/smart-contract-modules/rates-
                ↪ module
            To check this yourself, use the following rate calculation (example
                ↪ 8%):

            $ bc -l <<< 'scale=27; e( l(1.08)/(60 * 60 * 24 * 365) )'

            A table of rates can also be found at:
            https://ipfs.io/ipfs/QmefQMseb3AiTapiAKKexdKHig8wroKuZbmLtPLv4u2YwW

        @param _ilk    The ilk to update (ex. bytes32("ETH-A") )
        @param _rate   The accumulated rate (ex. 4% =>
            ↪ 1000000001243680656318820312)
        @param _doDrip `true` to accumulate stability fees for the collateral
    */
    function setIlkStabilityFee(bytes32 _ilk, uint256 _rate, bool _doDrip)
       ↪ public {
        require((_rate >= RAY) && (_rate <= RATES_ONE_HUNDRED_PCT));  // "
            ↪ LibDssExec/ilk-stability-fee-out-of-bounds"
        address _jug = jug();
        if (_doDrip) Drippable(_jug).drip(_ilk);

        setValue(_jug, _ilk, "duty", _rate);
    }
```

21.5.86 setLinearDecrease(_calc, _duration) X

```
    /**
        @dev Set the number of seconds from the start when the auction reaches
            ↪ zero price.
        @dev Abacus:LinearDecrease only.
        @param _calc      The address of the LinearDecrease pricing contract
        @param _duration Amount of time for auctions.
    */
    function setLinearDecrease(address _calc, uint256 _duration) public {
        setValue(_calc, "tau", _duration);
    }
```

21.5.87 setStairstepExponentialDecrease(_calc, _duration, _pct_bps) X

```
    /**
        @dev Set the number of seconds for each price step.
        @dev Abacus:StairstepExponentialDecrease only.
        @param _calc      The address of the StairstepExponentialDecrease pricing
            ↪  contract
        @param _duration Length of time between price drops [seconds]
        @param _pct_bps Per-step multiplicative factor in basis points. (ex. 99%
            ↪  == 9900)
    */
    function setStairstepExponentialDecrease(address _calc, uint256 _duration,
       ↪ uint256 _pct_bps) public {
        require(_pct_bps < BPS_ONE_HUNDRED_PCT); // DssExecLib/cut-too-high
        setValue(_calc, "cut", rdiv(_pct_bps, BPS_ONE_HUNDRED_PCT));
        setValue(_calc, "step", _duration);
    }
```

21.5.88 setExponentialDecrease(_calc, _pct_bps) X

```
    /**
        @dev Set the number of seconds for each price step. (99% cut = 1% price
            ↪ drop per step)
             Amounts will be converted to the correct internal precision.
        @dev Abacus:ExponentialDecrease only
        @param _calc     The address of the ExponentialDecrease pricing contract
        @param _pct_bps Per-step multiplicative factor in basis points. (ex. 99%
            ↪  == 9900)
    */
    function setExponentialDecrease(address _calc, uint256 _pct_bps) public {
        require(_pct_bps < BPS_ONE_HUNDRED_PCT); // DssExecLib/cut-too-high
        setValue(_calc, "cut", rdiv(_pct_bps, BPS_ONE_HUNDRED_PCT));
    }
```

21.5.89 whitelistOracleMedians(_oracle) X

```
    /**
        @dev Allows an oracle to read prices from its source feeds
        @param _oracle  An OSM or LP oracle contract
    */
    function whitelistOracleMedians(address _oracle) public {
        (bool ok, bytes memory data) = _oracle.call(abi.encodeWithSignature("
            ↪ orb0()"));
        if (ok) {
            // Token is an LP oracle
            address median0 = abi.decode(data, (address));
            addReaderToWhitelistCall(median0, _oracle);
            addReaderToWhitelistCall(OracleLike(_oracle).orb1(), _oracle);
        } else {
            // Standard OSM
            addReaderToWhitelistCall(OracleLike(_oracle).src(), _oracle);
        }
    }
```

21.5.90 addReaderToWhitelist(_oracle, _reader) X

```
    /**
        @dev Adds an address to the OSM or Median's reader whitelist, allowing
            ↪ the address to read prices.
        @param _oracle        Oracle Security Module (OSM) or Median core
            ↪ contract address
        @param _reader      Address to add to whitelist
    */
    function addReaderToWhitelist(address _oracle, address _reader) public {
        OracleLike(_oracle).kiss(_reader);
    }
```

21.5.91 removeReaderFromWhitelist(_oracle, _reader) X

```
    /**
        @dev Removes an address to the OSM or Median's reader whitelist,
            ↪ disallowing the address to read prices.
        @param _oracle      Oracle Security Module (OSM) or Median core contract
            ↪ address
        @param _reader      Address to remove from whitelist
    */
    function removeReaderFromWhitelist(address _oracle, address _reader) public
        ↪ {
        OracleLike(_oracle).diss(_reader);
    }
```

21.5.92 addReaderToWhitelistCall(_oracle, _reader) X

```
    /**
        @dev Adds an address to the OSM or Median's reader whitelist, allowing
            ↪ the address to read prices.
        @param _oracle  OSM or Median core contract address
        @param _reader  Address to add to whitelist
    */
    function addReaderToWhitelistCall(address _oracle, address _reader) public {
        (bool ok,) = _oracle.call(abi.encodeWithSignature("kiss(address)",
            ↪ _reader)); ok;
    }
```

21.5.93 removeReaderFromWhitelistCall(_oracle, _reader) X

```
    /**
        @dev Removes an address to the OSM or Median's reader whitelist,
            ↪ disallowing the address to read prices.
        @param _oracle  Oracle Security Module (OSM) or Median core contract
            ↪ address
        @param _reader  Address to remove from whitelist
    */
    function removeReaderFromWhitelistCall(address _oracle, address _reader)
        ↪ public {
        (bool ok,) = _oracle.call(abi.encodeWithSignature("diss(address)",
            ↪ _reader)); ok;
    }
```

21.5.94 setMedianWritersQuorum(_median, _minQuorum) X

```
    /**
        @dev Sets the minimum number of valid messages from whitelisted oracle
            ↪ feeds needed to update median price.
        @param _median     Median core contract address
        @param _minQuorum  Minimum number of valid messages from whitelisted
            ↪ oracle feeds needed to update median price (NOTE: MUST BE ODD
            ↪ NUMBER)
    */
    function setMedianWritersQuorum(address _median, uint256 _minQuorum) public
        ↪ {
        OracleLike(_median).setBar(_minQuorum);
    }
```

21.5.95 allowOSMFreeze(_osm, _ilk) X

```
    /**
        @dev Add OSM address to OSM mom, allowing it to be frozen by governance.
        @param _osm        Oracle Security Module (OSM) core contract address
        @param _ilk        Collateral type using OSM
    */
    function allowOSMFreeze(address _osm, bytes32 _ilk) public {
        MomLike(osmMom()).setOsm(_ilk, _osm);
    }
```

21.5.96 setD3MTargetInterestRate(_d3m, _pct_bps) X

```
/**
    @dev Sets the target rate threshold for a dai direct deposit module (d3m
      ↪ )
    @dev Aave: Targets the variable borrow rate
    @param _d3m     The address of the D3M contract
    @param _pct_bps Target rate in basis points. (ex. 4% == 400)
*/
function setD3MTargetInterestRate(address _d3m, uint256 _pct_bps) public {
    require(_pct_bps < BPS_ONE_HUNDRED_PCT); // DssExecLib/bar-too-high
    setValue(_d3m, "bar", rdiv(_pct_bps, BPS_ONE_HUNDRED_PCT));
}
```

21.5.97 addCollateralBase(_ilk, _gem, _join, _clip, _calc, _pip) X

```
/**
    @dev Performs basic functions and sanity checks to add a new collateral
      ↪ type to the MCD system
    @param _ilk      Collateral type key code [Ex. "ETH-A"]
    @param _gem      Address of token contract
    @param _join     Address of join adapter
    @param _clip     Address of liquidation agent
    @param _calc     Address of the pricing function
    @param _pip      Address of price feed
*/
function addCollateralBase(
    bytes32 _ilk,
    address _gem,
    address _join,
    address _clip,
    address _calc,
    address _pip
) public {
    // Sanity checks
    address _vat = vat();
    address _dog = dog();
    address _spotter = spotter();
    require(JoinLike(_join).vat() == _vat);     // "join-vat-not-match"
    require(JoinLike(_join).ilk() == _ilk);     // "join-ilk-not-match"
    require(JoinLike(_join).gem() == _gem);     // "join-gem-not-match"
    require(JoinLike(_join).dec() ==
                ERC20(_gem).decimals());        // "join-dec-not-match"
    require(ClipLike(_clip).vat() == _vat);     // "clip-vat-not-match"
    require(ClipLike(_clip).dog() == _dog);     // "clip-dog-not-match"
    require(ClipLike(_clip).ilk() == _ilk);     // "clip-ilk-not-match"
    require(ClipLike(_clip).spotter() == _spotter);  // "clip-ilk-not-match"

    // Set the token PIP in the Spotter
    setContract(spotter(), _ilk, "pip", _pip);

    // Set the ilk Clipper in the Dog
    setContract(_dog, _ilk, "clip", _clip);
    // Set vow in the clip
    setContract(_clip, "vow", vow());
    // Set the pricing function for the Clipper
    setContract(_clip, "calc", _calc);

    // Init ilk in Vat & Jug
    Initializable(_vat).init(_ilk);  // Vat
    Initializable(jug()).init(_ilk);  // Jug

    // Allow ilk Join to modify Vat registry
    authorize(_vat, _join);
    // Allow ilk Join to suck dai for keepers
    authorize(_vat, _clip);
    // Allow the ilk Clipper to reduce the Dog hole on deal()
```

```
        authorize(_dog, _clip);
        // Allow Dog to kick auctions in ilk Clipper
        authorize(_clip, _dog);
        // Allow End to yank auctions in ilk Clipper
        authorize(_clip, end());
        // Authorize the ESM to execute in the clipper
        authorize(_clip, esm());

        // Add new ilk to the IlkRegistry
        RegistryLike(reg()).add(_join);
    }
```

21.5.98 addNewCollateral(co) X

```
    // Complete collateral onboarding logic.
    function addNewCollateral(CollateralOpts memory co) public {
        // Add the collateral to the system.
        addCollateralBase(co.ilk, co.gem, co.join, co.clip, co.calc, co.pip);
        address clipperMom_ = clipperMom();

        if (!co.isLiquidatable) {
            // Disallow Dog to kick auctions in ilk Clipper
            setValue(co.clip, "stopped", 3);
        } else {
            // Grant ClipperMom access to the ilk Clipper
            authorize(co.clip, clipperMom_);
        }

        if(co.isOSM) { // If pip == OSM
            // Allow OsmMom to access to the TOKEN OSM
            authorize(co.pip, osmMom());
            if (co.whitelistOSM) { // If median is src in OSM
                // Whitelist OSM to read the Median data (only necessary if it
                    ↪ is the first time the token is being added to an ilk)
                whitelistOracleMedians(co.pip);
            }
            // Whitelist Spotter to read the OSM data (only necessary if it is
                ↪ the first time the token is being added to an ilk)
            addReaderToWhitelist(co.pip, spotter());
            // Whitelist Clipper on pip
            addReaderToWhitelist(co.pip, co.clip);
            // Allow the clippermom to access the feed
            addReaderToWhitelist(co.pip, clipperMom_);
            // Whitelist End to read the OSM data (only necessary if it is the
                ↪ first time the token is being added to an ilk)
            addReaderToWhitelist(co.pip, end());
            // Set TOKEN OSM in the OsmMom for new ilk
            allowOSMFreeze(co.pip, co.ilk);
        }
        // Increase the global debt ceiling by the ilk ceiling
        increaseGlobalDebtCeiling(co.ilkDebtCeiling);

        // Set the ilk debt ceiling
        setIlkDebtCeiling(co.ilk, co.ilkDebtCeiling);

        // Set the hole size
        setIlkMaxLiquidationAmount(co.ilk, co.maxLiquidationAmount);

        // Set the ilk dust
        setIlkMinVaultAmount(co.ilk, co.minVaultAmount);

        // Set the ilk liquidation penalty
        setIlkLiquidationPenalty(co.ilk, co.liquidationPenalty);

        // Set the ilk stability fee
        setIlkStabilityFee(co.ilk, co.ilkStabilityFee, true);
        // Set the auction starting price multiplier
        setStartingPriceMultiplicativeFactor(co.ilk, co.startingPriceFactor);

        // Set the amount of time before an auction resets.
        setAuctionTimeBeforeReset(co.ilk, co.auctionDuration);

        // Set the allowed auction drop percentage before reset
        setAuctionPermittedDrop(co.ilk, co.permittedDrop);

        // Set the ilk min collateralization ratio
        setIlkLiquidationRatio(co.ilk, co.liquidationRatio);

        // Set the price tolerance in the liquidation circuit breaker
        setLiquidationBreakerPriceTolerance(co.clip, co.breakerTolerance);

        // Set a flat rate for the keeper reward
        setKeeperIncentiveFlatRate(co.ilk, co.kprFlatReward);

        // Set the percentage of liquidation as keeper award
        setKeeperIncentivePercent(co.ilk, co.kprPctReward);

        // Update ilk spot value in Vat
        updateCollateralPrice(co.ilk);
    }
```

21.5.99 sendPaymentFromSurplusBuffer(_target, _amount) X

```
    /**
        @dev Send a payment in ERC20 DAI from the surplus buffer.
        @param _target The target address to send the DAI to.
        @param _amount The amount to send in DAI (ex. 10m DAI amount ==
            ↪ 10000000)
    */
    function sendPaymentFromSurplusBuffer(address _target, uint256 _amount)
        ↪ public {
        require(_amount < WAD);  // "LibDssExec/incorrect-ilk-line-precision"
        DssVat(vat()).suck(vow(), address(this), _amount * RAD);
        JoinLike(daiJoin()).exit(_target, _amount * WAD);
    }
```

21.5.100 linearInterpolation(_name, _target, _what, _startTime, _start, _end, _duration) X

```
    /**
        @dev Initiate linear interpolation on an administrative value over time.
        @param _name         The label for this lerp instance
        @param _target       The target contract
        @param _what         The target parameter to adjust
        @param _startTime    The time for this lerp
        @param _start        The start value for the target parameter
        @param _end          The end value for the target parameter
        @param _duration     The duration of the interpolation
    */
    function linearInterpolation(bytes32 _name, address _target, bytes32 _what,
        ↪ uint256 _startTime, uint256 _start, uint256 _end, uint256 _duration)
        ↪ public returns (address) {
        address lerp = LerpFactoryLike(lerpFab()).newLerp(_name, _target, _what,
            ↪ _startTime, _start, _end, _duration);
        Authorizable(_target).rely(lerp);
        LerpLike(lerp).tick();
        return lerp;
    }
```

21.5.101 linearInterpolation(_name, _target, _ilk, _what, _startTime, _start, _end, _duration) X

```
/**
    @dev Initiate linear interpolation on an administrative value over time.
    @param _name         The label for this lerp instance
    @param _target       The target contract
    @param _ilk          The ilk to target
    @param _what         The target parameter to adjust
    @param _startTime    The time for this lerp
    @param _start        The start value for the target parameter
    @param _end          The end value for the target parameter
    @param _duration     The duration of the interpolation
*/
function linearInterpolation(bytes32 _name, address _target, bytes32 _ilk,
   ↪ bytes32 _what, uint256 _startTime, uint256 _start, uint256 _end,
   ↪ uint256 _duration) public returns (address) {
    address lerp = LerpFactoryLike(lerpFab()).newIlkLerp(_name, _target,
       ↪ _ilk, _what, _startTime, _start, _end, _duration);
    Authorizable(_target).rely(lerp);
    LerpLike(lerp).tick();
    return lerp;
}
```

21.6 library SafeMath

```
library SafeMath {
}
```

21.6.1 mul(a, b)

```
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) {
            return 0;
        }
        uint256 c = a * b;
        assert(c / a == b);
        return c;
    }
```

21.6.2 div(a, b)

```
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        // assert(b > 0); // Solidity automatically throws when dividing by 0
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't
            ↪ hold
        return c;
    }
```

21.6.3 sub(a, b)

```
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        assert(b <= a);
        return a - b;
    }
```

21.6.4 add(a, b)

```
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        assert(c >= a);
        return c;
    }
```

21.7 library EnumerableSet

```
/**
 * @dev Library for managing
 * https://en.wikipedia.org/wiki/Set_(abstract_data_type)[sets] of primitive
 * types.
 *
 * Sets have the following properties:
 *
 * - Elements are added, removed, and checked for existence in constant time
 * (O(1)).
 * - Elements are enumerated in O(n). No guarantees are made on the ordering.
 *
 * ‘‘‘
 * contract Example {
 *     // Add the library methods
 *     using EnumerableSet for EnumerableSet.AddressSet;
 *
 *     // Declare a set state variable
 *     EnumerableSet.AddressSet private mySet;
 * }
 * ‘‘‘
 *
 * As of v3.3.0, sets of type ‘bytes32‘ (‘Bytes32Set‘), ‘address‘ (‘AddressSet‘)
 * and ‘uint256‘ (‘UintSet‘) are supported.
 */
library EnumerableSet {
}
```

21.7.1 struct EnumerableSet.Set

```
    // To implement this library for multiple types with as little code
    // repetition as possible, we write it in terms of a generic Set type with
    // bytes32 values.
    // The Set implementation uses private functions, and user-facing
    // implementations (such as AddressSet) are just wrappers around the
    // underlying Set.
    // This means that we can only create new EnumerableSets for types that fit
    // in bytes32.

    struct Set {
        // Storage of set values
        bytes32[] _values;
        // Position of the value in the ‘values‘ array, plus 1 because index 0
        // means a value is not in the set.
        mapping(bytes32 => uint256) _indexes;
    }
```

21.7.2 struct EnumerableSet.Bytes32Set

```
    // Bytes32Set

    struct Bytes32Set {
        Set _inner;
    }
```

21.7.3 struct EnumerableSet.AddressSet

```
    // AddressSet

    struct AddressSet {
        Set _inner;
    }
```

21.7.4 struct EnumerableSet.UintSet

```
// UintSet

struct UintSet {
    Set _inner;
}
```

21.7.5 _add(set, value)

```
/**
 * @dev Add a value to a set. O(1).
 *
 * Returns true if the value was added to the set, that is if it was not
 * already present.
 */
function _add(Set storage set, bytes32 value) private returns (bool) {
    if (!_contains(set, value)) {
        set._values.push(value);
        // The value is stored at length-1, but we add 1 to all indexes
        // and use 0 as a sentinel value
        set._indexes[value] = set._values.length;
        return true;
    } else {
        return false;
    }
}
```

21.7.6 _remove(set, value)

```
/**
 * @dev Removes a value from a set. O(1).
 *
 * Returns true if the value was removed from the set, that is if it was
 * present.
 */
function _remove(Set storage set, bytes32 value) private returns (bool) {
    // We read and store the value's index to prevent multiple reads from
        ↪ the same storage slot
    uint256 valueIndex = set._indexes[value];

    if (valueIndex != 0) {
        // Equivalent to contains(set, value)
        // To delete an element from the _values array in O(1), we swap the
            ↪ element to delete with the last one in
        // the array, and then remove the last element (sometimes called as
            ↪ 'swap and pop').
        // This modifies the order of the array, as noted in {at}.

        uint256 toDeleteIndex = valueIndex - 1;
        uint256 lastIndex = set._values.length - 1;

        if (lastIndex != toDeleteIndex) {
            bytes32 lastvalue = set._values[lastIndex];

            // Move the last value to the index where the value to delete is
            set._values[toDeleteIndex] = lastvalue;
            // Update the index for the moved value
            set._indexes[lastvalue] = valueIndex; // Replace lastvalue's
                ↪ index to valueIndex
        }

        // Delete the slot where the moved value was stored
        set._values.pop();

        // Delete the index for the deleted slot
```

```
            delete set._indexes[value];

            return true;
        } else {
            return false;
        }
    }
```

21.7.7 _contains(set, value)

```
    /**
     * @dev Returns true if the value is in the set. O(1).
     */
    function _contains(Set storage set, bytes32 value) private view returns (
        ↪ bool) {
        return set._indexes[value] != 0;
    }
```

21.7.8 _length(set)

```
    /**
     * @dev Returns the number of values on the set. O(1).
     */
    function _length(Set storage set) private view returns (uint256) {
        return set._values.length;
    }
```

21.7.9 _at(set, index)

```
    /**
     * @dev Returns the value stored at position `index` in the set. O(1).
     *
     * Note that there are no guarantees on the ordering of values inside the
     * array, and it may change when more values are added or removed.
     *
     * Requirements:
     *
     * - `index` must be strictly less than {length}.
     */
    function _at(Set storage set, uint256 index) private view returns (bytes32)
        ↪ {
        return set._values[index];
    }
```

21.7.10 _values(set)

```
    /**
     * @dev Return the entire set in an array
     *
     * WARNING: This operation will copy the entire storage to memory, which can
         ↪  be quite expensive. This is designed
     * to mostly be used by view accessors that are queried without any gas fees
         ↪ . Developers should keep in mind that
     * this function has an unbounded cost, and using it as part of a state-
         ↪ changing function may render the function
     * uncallable if the set grows to a point where copying to memory consumes
         ↪ too much gas to fit in a block.
     */
    function _values(Set storage set) private view returns (bytes32[] memory) {
        return set._values;
    }
```

21.7.11 add(set, value)

```
/**
 * @dev Add a value to a set. O(1).
 *
 * Returns true if the value was added to the set, that is if it was not
 * already present.
 */
function add(Bytes32Set storage set, bytes32 value) internal returns (bool)
    ↪ {
    return _add(set._inner, value);
}
```

21.7.12 remove(set, value)

```
/**
 * @dev Removes a value from a set. O(1).
 *
 * Returns true if the value was removed from the set, that is if it was
 * present.
 */
function remove(Bytes32Set storage set, bytes32 value) internal returns (
    ↪ bool) {
    return _remove(set._inner, value);
}
```

21.7.13 contains(set, value)

```
/**
 * @dev Returns true if the value is in the set. O(1).
 */
function contains(Bytes32Set storage set, bytes32 value) internal view
    ↪ returns (bool) {
    return _contains(set._inner, value);
}
```

21.7.14 length(set)

```
/**
 * @dev Returns the number of values in the set. O(1).
 */
function length(Bytes32Set storage set) internal view returns (uint256) {
    return _length(set._inner);
}
```

21.7.15 at(set, index)

```
/**
 * @dev Returns the value stored at position ‘index‘ in the set. O(1).
 *
 * Note that there are no guarantees on the ordering of values inside the
 * array, and it may change when more values are added or removed.
 *
 * Requirements:
 *
 * - ‘index‘ must be strictly less than {length}.
 */
function at(Bytes32Set storage set, uint256 index) internal view returns (
    ↪ bytes32) {
    return _at(set._inner, index);
}
```

21.7.16 values(set)

```
/**
 * @dev Return the entire set in an array
 *
 * WARNING: This operation will copy the entire storage to memory, which can
    ↪ be quite expensive. This is designed
 * to mostly be used by view accessors that are queried without any gas fees
    ↪ . Developers should keep in mind that
 * this function has an unbounded cost, and using it as part of a state-
    ↪ changing function may render the function
 * uncallable if the set grows to a point where copying to memory consumes
    ↪ too much gas to fit in a block.
 */
function values(Bytes32Set storage set) internal view returns (bytes32[]
    ↪ memory) {
    return _values(set._inner);
}
```

21.7.17 add(set, value)

```
/**
 * @dev Add a value to a set. O(1).
 *
 * Returns true if the value was added to the set, that is if it was not
 * already present.
 */
function add(AddressSet storage set, address value) internal returns (bool)
    ↪ {
    return _add(set._inner, bytes32(uint256(uint160(value))));
}
```

21.7.18 remove(set, value)

```
/**
 * @dev Removes a value from a set. O(1).
 *
 * Returns true if the value was removed from the set, that is if it was
 * present.
 */
function remove(AddressSet storage set, address value) internal returns (
    ↪ bool) {
    return _remove(set._inner, bytes32(uint256(uint160(value))));
}
```

21.7.19 contains(set, value)

```
/**
 * @dev Returns true if the value is in the set. O(1).
 */
function contains(AddressSet storage set, address value) internal view
    ↪ returns (bool) {
    return _contains(set._inner, bytes32(uint256(uint160(value))));
}
```

21.7.20 length(set)

```
/**
 * @dev Returns the number of values in the set. O(1).
 */
function length(AddressSet storage set) internal view returns (uint256) {
    return _length(set._inner);
}
```

21.7.21 at(set, index)

```
    /**
     * @dev Returns the value stored at position `index` in the set. O(1).
     *
     * Note that there are no guarantees on the ordering of values inside the
     * array, and it may change when more values are added or removed.
     *
     * Requirements:
     *
     * - `index` must be strictly less than {length}.
     */
    function at(AddressSet storage set, uint256 index) internal view returns (
        ↪ address) {
        return address(uint160(uint256(_at(set._inner, index))));
    }
```

21.7.22 values(set)

```
    /**
     * @dev Return the entire set in an array
     *
     * WARNING: This operation will copy the entire storage to memory, which can
         ↪  be quite expensive. This is designed
     * to mostly be used by view accessors that are queried without any gas fees
         ↪ . Developers should keep in mind that
     * this function has an unbounded cost, and using it as part of a state-
         ↪ changing function may render the function
     * uncallable if the set grows to a point where copying to memory consumes
         ↪ too much gas to fit in a block.
     */
    function values(AddressSet storage set) internal view returns (address[]
        ↪ memory) {
        bytes32[] memory store = _values(set._inner);
        address[] memory result;

        assembly {
            result := store
        }

        return result;
    }
```

21.7.23 add(set, value)

```
    /**
     * @dev Add a value to a set. O(1).
     *
     * Returns true if the value was added to the set, that is if it was not
     * already present.
     */
    function add(UintSet storage set, uint256 value) internal returns (bool) {
        return _add(set._inner, bytes32(value));
    }
```

21.7.24 remove(set, value)

```
    /**
     * @dev Removes a value from a set. O(1).
     *
     * Returns true if the value was removed from the set, that is if it was
     * present.
     */
    function remove(UintSet storage set, uint256 value) internal returns (bool)
        ↪ {
        return _remove(set._inner, bytes32(value));
    }
```

21.7.25 contains(set, value)

```
    /**
     * @dev Returns true if the value is in the set. O(1).
     */
    function contains(UintSet storage set, uint256 value) internal view returns
        ↪ (bool) {
        return _contains(set._inner, bytes32(value));
    }
```

21.7.26 length(set)

```
    /**
     * @dev Returns the number of values on the set. O(1).
     */
    function length(UintSet storage set) internal view returns (uint256) {
        return _length(set._inner);
    }
```

21.7.27 at(set, index)

```
    /**
     * @dev Returns the value stored at position ‘index‘ in the set. O(1).
     *
     * Note that there are no guarantees on the ordering of values inside the
     * array, and it may change when more values are added or removed.
     *
     * Requirements:
     *
     * - ‘index‘ must be strictly less than {length}.
     */
    function at(UintSet storage set, uint256 index) internal view returns (
        ↪ uint256) {
        return uint256(_at(set._inner, index));
    }
```

21.7.28 values(set)

```
    /**
     * @dev Return the entire set in an array
     *
     * WARNING: This operation will copy the entire storage to memory, which can
         ↪  be quite expensive. This is designed
     * to mostly be used by view accessors that are queried without any gas fees
         ↪ . Developers should keep in mind that
     * this function has an unbounded cost, and using it as part of a state-
         ↪ changing function may render the function
     * uncallable if the set grows to a point where copying to memory consumes
         ↪ too much gas to fit in a block.
     */
    function values(UintSet storage set) internal view returns (uint256[] memory
        ↪ ) {
        bytes32[] memory store = _values(set._inner);
        uint256[] memory result;

        assembly {
            result := store
        }
        return result;
    }
```

21.8 library Address

```
/**
 * @dev Collection of functions related to the address type
 */
library Address {
}
```

21.8.1 isContract(account)

```
    /**
     * @dev Returns true if `account` is a contract.
     *
     * [IMPORTANT]
     * ====
     * It is unsafe to assume that an address for which this function returns
     * false is an externally-owned account (EOA) and not a contract.
     *
     * Among others, `isContract` will return false for the following
     * types of addresses:
     *
     *  - an externally-owned account
     *  - a contract in construction
     *  - an address where a contract will be created
     *  - an address where a contract lived, but was destroyed
     * ====
     */
    function isContract(address account) internal view returns (bool) {
        // This method relies on extcodesize, which returns 0 for contracts in
        // construction, since the code is only stored at the end of the
        // constructor execution.

        uint256 size;
        // solhint-disable-next-line no-inline-assembly
        assembly { size := extcodesize(account) }
        return size > 0;
    }
```

21.8.2 sendValue(recipient, amount)

```
    /**
     * @dev Replacement for Solidity's `transfer`: sends `amount` wei to
     * `recipient`, forwarding all available gas and reverting on errors.
     *
     * https://eips.ethereum.org/EIPS/eip-1884[EIP1884] increases the gas cost
     * of certain opcodes, possibly making contracts go over the 2300 gas limit
     * imposed by `transfer`, making them unable to receive funds via
     * `transfer`. {sendValue} removes this limitation.
     *
     * https://diligence.consensys.net/posts/2019/09/stop-using-soliditys-
       ↪ transfer-now/[Learn more].
     *
     * IMPORTANT: because control is transferred to `recipient`, care must be
     * taken to not create reentrancy vulnerabilities. Consider using
     * {ReentrancyGuard} or the
     * https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html#
       ↪ use-the-checks-effects-interactions-pattern[checks-effects-
       ↪ interactions pattern].
     */
    function sendValue(address payable recipient, uint256 amount) internal {
        require(address(this).balance >= amount, "Address: insufficient balance"
            ↪ );

        // solhint-disable-next-line avoid-low-level-calls, avoid-call-value
        (bool success, ) = recipient.call{ value: amount }("");
        require(success, "Address: unable to send value, recipient may have
            ↪ reverted");
    }
```

21.8.3 functionCall(target, data)

```
    /**
     * @dev Performs a Solidity function call using a low level ‘call‘. A
     * plain‘call‘ is an unsafe replacement for a function call: use this
     * function instead.
     *
     * If ‘target‘ reverts with a revert reason, it is bubbled up by this
     * function (like regular Solidity function calls).
     *
     * Returns the raw returned data. To convert to the expected return value,
     * use https://solidity.readthedocs.io/en/latest/units-and-global-variables.
        ↪ html?highlight=abi.decode#abi-encoding-and-decoding-functions[‘abi.
        ↪ decode‘].
     *
     * Requirements:
     *
     * - ‘target‘ must be a contract.
     * - calling ‘target‘ with ‘data‘ must not revert.
     *
     * _Available since v3.1._
     */
    function functionCall(address target, bytes memory data) internal returns (
        ↪ bytes memory) {
      return functionCall(target, data, "Address: low-level call failed");
    }
```

21.8.4 functionCall(target, data, errorMessage)

```
    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-}[‘functionCall‘],
        ↪ but with
     * ‘errorMessage‘ as a fallback revert reason when ‘target‘ reverts.
     *
     * _Available since v3.1._
     */
    function functionCall(address target, bytes memory data, string memory
        ↪ errorMessage) internal returns (bytes memory) {
        return functionCallWithValue(target, data, 0, errorMessage);
    }
```

21.8.5 functionCallWithValue(target, data, value)

```
    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-}[‘functionCall‘],
     * but also transferring ‘value‘ wei to ‘target‘.
     *
     * Requirements:
     *
     * - the calling contract must have an ETH balance of at least ‘value‘.
     * - the called Solidity function must be ‘payable‘.
     *
     * _Available since v3.1._
     */
    function functionCallWithValue(address target, bytes memory data, uint256
        ↪ value) internal returns (bytes memory) {
        return functionCallWithValue(target, data, value, "Address: low-level
            ↪ call with value failed");
    }
```

21.8.6 functionCallWithValue(target, data, value, errorMessage)

```
    /**
     * @dev Same as {xref-Address-functionCallWithValue-address-bytes-uint256
        ↪ -}[`functionCallWithValue`], but
     * with `errorMessage` as a fallback revert reason when `target` reverts.
     *
     * _Available since v3.1._
     */
    function functionCallWithValue(address target, bytes memory data, uint256
       ↪ value, string memory errorMessage) internal returns (bytes memory) {
        require(address(this).balance >= value, "Address: insufficient balance
           ↪ for call");
        require(isContract(target), "Address: call to non-contract");

        // solhint-disable-next-line avoid-low-level-calls
        (bool success, bytes memory returndata) = target.call{ value: value }(
           ↪ data);
        return _verifyCallResult(success, returndata, errorMessage);
    }
```

21.8.7 functionStaticCall(target, data)

```
    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
     * but performing a static call.
     *
     * _Available since v3.3._
     */
    function functionStaticCall(address target, bytes memory data) internal view
       ↪ returns (bytes memory) {
        return functionStaticCall(target, data, "Address: low-level static call
           ↪ failed");
    }
```

21.8.8 functionStaticCall(target, data, errorMessage)

```
    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-string-}[`
        ↪ functionCall`],
     * but performing a static call.
     *
     * _Available since v3.3._
     */
    function functionStaticCall(address target, bytes memory data, string memory
       ↪ errorMessage) internal view returns (bytes memory) {
        require(isContract(target), "Address: static call to non-contract");

        // solhint-disable-next-line avoid-low-level-calls
        (bool success, bytes memory returndata) = target.staticcall(data);
        return _verifyCallResult(success, returndata, errorMessage);
    }
```

21.8.9 functionDelegateCall(target, data)

```
    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
     * but performing a delegate call.
     *
     * _Available since v3.4._
     */
    function functionDelegateCall(address target, bytes memory data) internal
       ↪ returns (bytes memory) {
        return functionDelegateCall(target, data, "Address: low-level delegate
            ↪ call failed");
    }
```

21.8.10 functionDelegateCall(target, data, errorMessage)

```
    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-string-}[`
        ↪ functionCall`],
     * but performing a delegate call.
     *
     * _Available since v3.4._
     */
    function functionDelegateCall(address target, bytes memory data, string
        ↪ memory errorMessage) internal returns (bytes memory) {
        require(isContract(target), "Address: delegate call to non-contract");

        // solhint-disable-next-line avoid-low-level-calls
        (bool success, bytes memory returndata) = target.delegatecall(data);
        return _verifyCallResult(success, returndata, errorMessage);
    }
```

21.8.11 _verifyCallResult(success, returndata, errorMessage)

```
    function _verifyCallResult(bool success, bytes memory returndata, string
        ↪ memory errorMessage) private pure returns(bytes memory) {
        if (success) {
            return returndata;
        } else {
            // Look for revert reason and bubble it up if present
            if (returndata.length > 0) {
                // The easiest way to bubble the revert reason is using memory
                    ↪ via assembly

                // solhint-disable-next-line no-inline-assembly
                assembly {
                    let returndata_size := mload(returndata)
                    revert(add(32, returndata), returndata_size)
                }
            } else {
                revert(errorMessage);
            }
        }
    }
```

21.9 library GodMode

```
library GodMode {

    address constant public VM_ADDR = address(bytes20(uint160(uint256(keccak256(
        ↪ 'hevm cheat code')))));

}
```

21.9.1 vm()

```
    function vm() internal pure returns (Vm) {
        return Vm(VM_ADDR);
    }
```

21.9.2 setWard(base, target, val)

```
    /// @dev Set the ward for `base` for the specified `target`
    /// Note this only works for contracts compiled under Solidity. Vyper
        ↪ contracts use a different storage structure for maps.
    /// See https://twitter.com/msolomon44/status/1420137730009300992?t=
        ↪ WO2052xM3AzUCL7o7Pfkow&s=19
    function setWard(address base, address target, uint256 val) internal {

        // Edge case - ward is already set
        if (WardsAbstract(base).wards(target) == val) return;

        for (int i = 0; i < 100; i++) {
            // Scan the storage for the ward storage slot
            bytes32 prevValue = vm().load(
                address(base),
                keccak256(abi.encode(target, uint256(i)))
            );
            vm().store(
                address(base),
                keccak256(abi.encode(target, uint256(i))),
                bytes32(uint256(val))
            );
            if (WardsAbstract(base).wards(target) == val) {
                // Found it
                return;
            } else {
                // Keep going after restoring the original value
                vm().store(
                    address(base),
                    keccak256(abi.encode(target, uint256(i))),
                    prevValue
                );
            }
        }

        // We have failed if we reach here
        revert("Could not give auth access");
    }
```

21.9.3 setWard(base, target, val)

```
    /// @dev Set the ward for `base` for the specified `target`
    /// Note this only works for contracts compiled under Solidity. Vyper
        ↪ contracts use a different storage structure for maps.
    /// See https://twitter.com/msolomon44/status/1420137730009300992?t=
        ↪ WO2052xM3AzUCL7o7Pfkow&s=19
    function setWard(WardsAbstract base, address target, uint256 val) internal {
        setWard(address(base), target, val);
    }
```

21.9.4 setWard(base, target, val)

```
    /// @dev Set the ward for `base` for the specified `target`
    /// Note this only works for contracts compiled under Solidity. Vyper
        ↪ contracts use a different storage structure for maps.
    /// See https://twitter.com/msolomon44/status/1420137730009300992?t=
        ↪ WO2052xM3AzUCL7o7Pfkow&s=19
    function setWard(VatAbstract base, address target, uint256 val) internal {
        setWard(address(base), target, val);
    }
```

21.9.5 setBalance(token, who, amount)

```
    /// @dev Sets the balance for `who` to `amount` for `token`.
    function setBalance(address token, address who, uint256 amount) internal {
        // Edge case - balance is already set for some reason
        if (DSTokenAbstract(token).balanceOf(who) == amount) return;

        for (uint256 i = 0; i < 200; i++) {
            // Scan the storage for the solidity-style balance storage slot
            {
                bytes32 prevValue = vm().load(
                    token,
                    keccak256(abi.encode(who, uint256(i)))
                );
                vm().store(
                    token,
                    keccak256(abi.encode(who, uint256(i))),
                    bytes32(amount)
                );
                if (DSTokenAbstract(token).balanceOf(who) == amount) {
                    // Found it
                    return;
                } else {
                    // Keep going after restoring the original value
                    vm().store(
                        token,
                        keccak256(abi.encode(who, uint256(i))),
                        prevValue
                    );
                }
            }

            // Vyper-style storage layout for maps
            {
                bytes32 prevValue = vm().load(
                    token,
                    keccak256(abi.encode(uint256(i), who))
                );

                vm().store(
                    token,
                    keccak256(abi.encode(uint256(i), who)),
                    bytes32(amount)
                );
                if (DSTokenAbstract(token).balanceOf(who) == amount) {
                    // Found it
                    return;
                } else {
                    // Keep going after restoring the original value
                    vm().store(
                        token,
                        keccak256(abi.encode(uint256(i), who)),
                        prevValue
                    );
                }
            }
        }

        // We have failed if we reach here
        revert("Could not give tokens");
    }
```

21.9.6 setBalance(token, who, amount)

```
    /// @dev Sets the balance for `who` to `amount` for `token`.
    function setBalance(DSTokenAbstract token, address who, uint256 amount)
        ↪ internal {
        setBalance(address(token), who, amount);
    }
```

21.9.7 setBalance(token, who, amount)

```
    /// @dev Sets the balance for `who` to `amount` for `token`.
    function setBalance(DaiAbstract token, address who, uint256 amount) internal
        ↪ {
        setBalance(address(token), who, amount);
    }
```