

MakerDAO

Auto-generated by code2pdf

February 10, 2023

Contents

References index	35
------------------	----

1 Core	223
1.1 <code>contract</code> Dai	223
1.1.1 <code>modifier</code> <code>auth()</code>	223
1.1.2 <code>rely(guy)</code> X a	223
1.1.3 <code>deny(guy)</code> X a	223
1.1.4 <code>add(x, y)</code>	224
1.1.5 <code>sub(x, y)</code>	224
1.1.6 <code>constructor(chainId_)</code> X	224
1.1.7 <code>transfer(dst, wad)</code> X	224
1.1.8 <code>transferFrom(src, dst, wad)</code> X	224
1.1.9 <code>mint(usr, wad)</code> X a	224
1.1.10 <code>burn(usr, wad)</code> X	225
1.1.11 <code>approve(usr, wad)</code> X	225
1.1.12 <code>push(usr, wad)</code> X	225
1.1.13 <code>pull(usr, wad)</code> X	225
1.1.14 <code>move(src, dst, wad)</code> X	225
1.1.15 <code>permit(holder, spender, nonce, expiry, allowed, v, r, s)</code> X	225
1.2 <code>contract</code> Spotter	227
1.2.1 <code>struct</code> Spotter.Ilk	227
1.2.2 <code>modifier</code> <code>auth()</code>	227
1.2.3 <code>rely(guy)</code> X a	227
1.2.4 <code>deny(guy)</code> X a	227
1.2.5 <code>constructor(vat_)</code> X	227
1.2.6 <code>mul(x, y)</code>	228
1.2.7 <code>rdiv(x, y)</code>	228
1.2.8 <code>file(ilk, what, pip_)</code> X a	228
1.2.9 <code>file(what, data)</code> X a	228
1.2.10 <code>file(ilk, what, data)</code> X a	228
1.2.11 <code>poke(ilk)</code> X	228
1.2.12 <code>cage()</code> X a	228
1.3 <code>contract</code> Vat	229
1.3.1 <code>struct</code> Vat.Ilk	229
1.3.2 <code>struct</code> Vat.Urn	229
1.3.3 <code>modifier</code> <code>auth()</code>	229
1.3.4 <code>rely(usr)</code> X a	229
1.3.5 <code>deny(usr)</code> X a	229
1.3.6 <code>hope(usr)</code> X	230
1.3.7 <code>nope(usr)</code> X	230
1.3.8 <code>wish(bit, usr)</code>	230
1.3.9 <code>constructor()</code> X	230
1.3.10 <code>_add(x, y)</code>	230
1.3.11 <code>_sub(x, y)</code>	230
1.3.12 <code>_mul(x, y)</code>	230
1.3.13 <code>_add(x, y)</code>	230
1.3.14 <code>_sub(x, y)</code>	231
1.3.15 <code>_mul(x, y)</code>	231
1.3.16 <code>init(ilk)</code> X a	231
1.3.17 <code>file(what, data)</code> X a	231
1.3.18 <code>file(ilk, what, data)</code> X a	231
1.3.19 <code>cage()</code> X a	231

1.3.20	slip(ilk, usr, wad) X a	231
1.3.21	flux(ilk, src, dst, wad) X	231
1.3.22	move(src, dst, rad) X	232
1.3.23	either(x, y)	232
1.3.24	both(x, y)	232
1.3.25	frob(i, u, v, w, dink, dart) X	232
1.3.26	fork(ilk, src, dst, dink, dart) X	233
1.3.27	grab(i, u, v, w, dink, dart) X a	233
1.3.28	heal(rad) X	233
1.3.29	suck(u, v, rad) X a	234
1.3.30	fold(i, u, rate) X a	234
2	Liquidations	235
2.1	contract LinearDecrease	235
2.1.1	modifier auth()	235
2.1.2	rely(usr) X a	235
2.1.3	deny(usr) X a	235
2.1.4	constructor() X	235
2.1.5	file(what, data) X a	236
2.1.6	add(x, y)	236
2.1.7	mul(x, y)	236
2.1.8	rmul(x, y)	236
2.1.9	price(top, dur)	236
2.2	contract StairstepExponentialDecrease	237
2.2.1	modifier auth()	237
2.2.2	rely(usr) X a	237
2.2.3	deny(usr) X a	237
2.2.4	constructor() X	237
2.2.5	file(what, data) X a	237
2.2.6	rmul(x, y)	238
2.2.7	rpow(x, n, b)	238
2.2.8	price(top, dur)	238
2.3	contract ExponentialDecrease	239
2.3.1	modifier auth()	239
2.3.2	rely(usr) X a	239
2.3.3	deny(usr) X a	239
2.3.4	constructor() X	239
2.3.5	file(what, data) X a	239
2.3.6	rmul(x, y)	240
2.3.7	rpow(x, n, b)	240
2.3.8	price(top, dur)	240
2.4	contract Clipper	241
2.4.1	struct Clipper.Sale	242
2.4.2	modifier auth()	242
2.4.3	modifier lock()	242
2.4.4	modifier isStopped(level)	242
2.4.5	rely(usr) X a	242
2.4.6	deny(usr) X a	242
2.4.7	constructor(vat_, spotter_, dog_, ilk_) X	243
2.4.8	file(what, data) X a	243
2.4.9	file(what, data) X a	243
2.4.10	min(x, y)	243
2.4.11	add(x, y)	243
2.4.12	sub(x, y)	243
2.4.13	mul(x, y)	244
2.4.14	wmul(x, y)	244
2.4.15	rmul(x, y)	244
2.4.16	rdiv(x, y)	244
2.4.17	getFeedPrice()	244
2.4.18	kick(tab, lot, usr, kpr) X a	244
2.4.19	redo(id, kpr) X	245
2.4.20	take(id, amt, max, who, data) X	246
2.4.21	remove(id)	247

2.4.22	count()	248
2.4.23	list()	248
2.4.24	getStatus(id)	248
2.4.25	status(tic, top)	248
2.4.26	upchost() X	248
2.4.27	yank(id) X a	249
2.5	contract Dog	251
2.5.1	struct Dog.Ilkk	251
2.5.2	modifier auth()	251
2.5.3	rely(usr) X a	252
2.5.4	deny(usr) X a	252
2.5.5	constructor(vat_) X	252
2.5.6	min(x, y)	252
2.5.7	add(x, y)	252
2.5.8	sub(x, y)	252
2.5.9	mul(x, y)	252
2.5.10	file(what, data) X a	252
2.5.11	file(what, data) X a	252
2.5.12	file(ilkk, what, data) X a	253
2.5.13	file(ilkk, what, clip) X a	253
2.5.14	chop(ilkk)	253
2.5.15	bark(ilkk, urn, kpr) X	253
2.5.16	digs(ilkk, rad) X a	255
2.5.17	cage() X a	255
2.6	contract ClipperMom	257
2.6.1	modifier onlyOwner()	257
2.6.2	modifier auth()	257
2.6.3	constructor(spotter_) X	257
2.6.4	add(x, y)	257
2.6.5	mul(x, y)	257
2.6.6	rmul(x, y)	257
2.6.7	isAuthorized(src, sig)	258
2.6.8	getPrices(clip)	258
2.6.9	setOwner(owner_) X	258
2.6.10	setAuthority(authority_) X	258
2.6.11	setPriceTolerance(clip, value) X	258
2.6.12	setBreaker(clip, level, delay) X a	258
2.6.13	tripBreaker(clip) X	259
3	System Stabilizer	261
3.1	contract Flapper	261
3.1.1	struct Flapper.Bid	261
3.1.2	modifier auth()	262
3.1.3	rely(usr) X a	262
3.1.4	deny(usr) X a	262
3.1.5	constructor(vat_, gem_) X	262
3.1.6	add(x, y)	262
3.1.7	add256(x, y)	262
3.1.8	sub(x, y)	262
3.1.9	mul(x, y)	262
3.1.10	file(what, data) X a	263
3.1.11	kick.lot, bid) X a	263
3.1.12	tick(id) X	263
3.1.13	tend(id, lot, bid) X	263
3.1.14	deal(id) X	264
3.1.15	cage(rad) X a	264
3.1.16	yank(id) X	264
3.2	contract Flopper	265
3.2.1	struct Flopper.Bid	265
3.2.2	modifier auth()	265
3.2.3	rely(usr) X a	265
3.2.4	deny(usr) X a	266
3.2.5	constructor(vat_, gem_) X	266

3.2.6	add(x, y)	266
3.2.7	mul(x, y)	266
3.2.8	min(x, y)	266
3.2.9	file(what, data) X a	266
3.2.10	kick(gal, lot, bid) X a	266
3.2.11	tick(id) X	267
3.2.12	dent(id, lot, bid) X	267
3.2.13	deal(id) X	267
3.2.14	cage() X a	267
3.2.15	yank(id) X	267
3.3	contract Vow	269
3.3.1	modifier auth()	269
3.3.2	rely(usr) X a	269
3.3.3	deny(usr) X a	269
3.3.4	constructor(vat_, flapper_, flopper_) X	269
3.3.5	add(x, y)	269
3.3.6	sub(x, y)	270
3.3.7	min(x, y)	270
3.3.8	file(what, data) X a	270
3.3.9	file(what, data) X a	270
3.3.10	fess(tab) X a	270
3.3.11	flog(era) X	270
3.3.12	heal(rad) X	270
3.3.13	kiss(rad) X	271
3.3.14	flop() X	271
3.3.15	flap() X	271
3.3.16	cage() X a	271
4	Rates	273
4.1	contract Jug	273
4.1.1	struct Jug.Ilk	273
4.1.2	modifier auth()	273
4.1.3	rely(usr) X a	273
4.1.4	deny(usr) X a	273
4.1.5	constructor(vat_) X	273
4.1.6	_rpow(x, n, b)	274
4.1.7	_add(x, y)	274
4.1.8	_diff(x, y)	274
4.1.9	_rmul(x, y)	274
4.1.10	init(ilk) X a	274
4.1.11	file(ilk, what, data) X a	275
4.1.12	file(what, data) X a	275
4.1.13	file(what, data) X a	275
4.1.14	drip(ilk) X	275
4.2	contract Pot	277
4.2.1	modifier auth()	277
4.2.2	rely(guy) X a	277
4.2.3	deny(guy) X a	277
4.2.4	constructor(vat_) X	277
4.2.5	_rpow(x, n, base)	277
4.2.6	_rmul(x, y)	278
4.2.7	_add(x, y)	278
4.2.8	_sub(x, y)	278
4.2.9	_mul(x, y)	278
4.2.10	file(what, data) X a	278
4.2.11	file(what, addr) X a	278
4.2.12	cage() X a	279
4.2.13	drip() X	279
4.2.14	join(wad) X	279
4.2.15	exit(wad) X	279
5	OSM	281
5.1	contract Median	281

5.1.1	<code>modifier</code> <code>note()</code> <code>[LibNote]</code>	281
5.1.2	<code>modifier</code> <code>auth()</code>	282
5.1.3	<code>modifier</code> <code>toll()</code>	282
5.1.4	<code>rely(usr)</code> <code>X a</code>	282
5.1.5	<code>deny(usr)</code> <code>X a</code>	282
5.1.6	<code>constructor()</code> <code>X</code>	282
5.1.7	<code>read()</code>	282
5.1.8	<code>peek()</code>	282
5.1.9	<code>recover(val_, age_, v, r, s)</code>	283
5.1.10	<code>poke(val_, age_, v, r, s)</code> <code>X</code>	283
5.1.11	<code>lift(a)</code> <code>X a</code>	283
5.1.12	<code>drop(a)</code> <code>X a</code>	283
5.1.13	<code>setBar(bar_)</code> <code>X a</code>	284
5.1.14	<code>kiss(a)</code> <code>X a</code>	284
5.1.15	<code>diss(a)</code> <code>X a</code>	284
5.1.16	<code>kiss(a)</code> <code>X a</code>	284
5.1.17	<code>diss(a)</code> <code>X a</code>	284
5.2	<code>contract</code> <code>OSM</code>	285
5.2.1	<code>struct</code> <code>OSM.Feed</code>	285
5.2.2	<code>modifier</code> <code>note()</code> <code>[LibNote(2)]</code>	285
5.2.3	<code>modifier</code> <code>auth()</code>	286
5.2.4	<code>modifier</code> <code>stoppable()</code>	286
5.2.5	<code>modifier</code> <code>toll()</code>	286
5.2.6	<code>rely(usr)</code> <code>X a</code>	286
5.2.7	<code>deny(usr)</code> <code>X a</code>	286
5.2.8	<code>add(x, y)</code>	286
5.2.9	<code>constructor(src_)</code> <code>X</code>	286
5.2.10	<code>stop()</code> <code>X a</code>	286
5.2.11	<code>start()</code> <code>X a</code>	286
5.2.12	<code>change(src_)</code> <code>X a</code>	287
5.2.13	<code>era()</code>	287
5.2.14	<code>prev(ts)</code>	287
5.2.15	<code>step(ts)</code> <code>X a</code>	287
5.2.16	<code>void()</code> <code>X a</code>	287
5.2.17	<code>pass()</code>	287
5.2.18	<code>poke()</code> <code>X</code>	287
5.2.19	<code>peek()</code>	287
5.2.20	<code>peep()</code>	288
5.2.21	<code>read()</code>	288
5.2.22	<code>kiss(a)</code> <code>X a</code>	288
5.2.23	<code>diss(a)</code> <code>X a</code>	288
5.2.24	<code>kiss(a)</code> <code>X a</code>	288
5.2.25	<code>diss(a)</code> <code>X a</code>	288
5.3	<code>contract</code> <code>Value</code>	289
5.3.1	<code>peek()</code>	289
6	Emergency Shutdown	291
6.1	<code>contract</code> <code>Cure</code>	291
6.1.1	<code>modifier</code> <code>auth()</code>	291
6.1.2	<code>_add(x, y)</code>	291
6.1.3	<code>_sub(x, y)</code>	291
6.1.4	<code>constructor()</code> <code>X</code>	292
6.1.5	<code>tCount()</code>	292
6.1.6	<code>list()</code>	292
6.1.7	<code>tell()</code>	292
6.1.8	<code>rely(usr)</code> <code>X a</code>	292
6.1.9	<code>deny(usr)</code> <code>X a</code>	292
6.1.10	<code>file(what, data)</code> <code>X a</code>	292
6.1.11	<code>lift(src)</code> <code>X a</code>	293
6.1.12	<code>drop(src)</code> <code>X a</code>	293
6.1.13	<code>cage()</code> <code>X a</code>	293
6.1.14	<code>load(src)</code> <code>X</code>	293
6.2	<code>contract</code> <code>End</code>	295

6.2.1	<code>modifier</code> <code>auth()</code>	297
6.2.2	<code>rely(usr)</code> <code>X a</code>	297
6.2.3	<code>deny(usr)</code> <code>X a</code>	297
6.2.4	<code>constructor()</code> <code>X</code>	297
6.2.5	<code>add(x, y)</code>	297
6.2.6	<code>sub(x, y)</code>	298
6.2.7	<code>mul(x, y)</code>	298
6.2.8	<code>min(x, y)</code>	298
6.2.9	<code>rmul(x, y)</code>	298
6.2.10	<code>wdiv(x, y)</code>	298
6.2.11	<code>file(what, data)</code> <code>X a</code>	298
6.2.12	<code>file(what, data)</code> <code>X a</code>	298
6.2.13	<code>cage()</code> <code>X a</code>	299
6.2.14	<code>cage(ilk)</code> <code>X</code>	299
6.2.15	<code>snip(ilk, id)</code> <code>X</code>	299
6.2.16	<code>skip(ilk, id)</code> <code>X</code>	299
6.2.17	<code>skim(ilk, urn)</code> <code>X</code>	300
6.2.18	<code>free(ilk)</code> <code>X</code>	300
6.2.19	<code>thaw()</code> <code>X</code>	300
6.2.20	<code>flow(ilk)</code> <code>X</code>	300
6.2.21	<code>pack(wad)</code> <code>X</code>	301
6.2.22	<code>cash(ilk, wad)</code> <code>X</code>	301
6.3	<code>contract</code> <code>ESM</code>	303
6.3.1	<code>modifier</code> <code>auth()</code>	303
6.3.2	<code>constructor(gem_, end_, proxy_, min_)</code> <code>X</code>	303
6.3.3	<code>revokesGovernanceAccess()</code>	303
6.3.4	<code>add(x, y)</code>	303
6.3.5	<code>rely(usr)</code> <code>X a</code>	304
6.3.6	<code>deny(usr)</code> <code>X a</code>	304
6.3.7	<code>file(what, data)</code> <code>X a</code>	304
6.3.8	<code>file(what, data)</code> <code>X a</code>	304
6.3.9	<code>cage()</code> <code>X a</code>	304
6.3.10	<code>fire()</code> <code>X</code>	304
6.3.11	<code>denyProxy(target)</code> <code>X</code>	305
6.3.12	<code>join(wad)</code> <code>X</code>	305
6.3.13	<code>burn()</code> <code>X</code>	305
7	Exec Lib	307
7.1	<code>contract</code> <code>DssAction</code>	307
7.1.1	<code>modifier</code> <code>limited()</code>	307
7.1.2	<code>officeHours()</code>	307
7.1.3	<code>execute()</code> <code>X</code>	307
7.1.4	<code>nextCastTime(eta)</code>	308
7.2	<code>contract</code> <code>DssExec</code>	309
7.2.1	<code>description()</code>	309
7.2.2	<code>officeHours()</code>	309
7.2.3	<code>nextCastTime()</code>	309
7.2.4	<code>constructor(_expiration, _spellAction)</code> <code>X</code>	309
7.2.5	<code>schedule()</code> <code>X</code>	309
7.2.6	<code>cast()</code> <code>X</code>	310
8	Direct deposit	311
8.1	<code>contract</code> <code>D3MAavePlan</code>	311
8.1.1	<code>modifier</code> <code>auth()</code>	311
8.1.2	<code>constructor(dai_, pool_)</code> <code>X</code>	311
8.1.3	<code>_rmul(x, y)</code>	312
8.1.4	<code>_rdiv(x, y)</code>	312
8.1.5	<code>rely(usr)</code> <code>X a</code>	312
8.1.6	<code>deny(usr)</code> <code>X a</code>	312
8.1.7	<code>file(what, data)</code> <code>X a</code>	312
8.1.8	<code>file(what, data)</code> <code>X a</code>	312
8.1.9	<code>_calculateTargetSupply(targetInterestRate, totalDebt)</code>	313
8.1.10	<code>getTargetAssets(currentAssets)</code>	313

8.1.11	active()	314
8.1.12	disable() X	314
8.2	contract D3MCompoundPlan	315
8.2.1	modifier auth()	315
8.2.2	constructor(cDai_) X	315
8.2.3	_wmul(x, y)	315
8.2.4	_wdiv(x, y)	316
8.2.5	rely(usr) X a	316
8.2.6	deny(usr) X a	316
8.2.7	file(what, data) X a	316
8.2.8	file(what, addr, data) X a	316
8.2.9	_calculateTargetSupply(targetInterestRate, borrows)	316
8.2.10	getTargetAssets(currentAssets)	317
8.2.11	active()	317
8.2.12	disable() X	317
8.3	contract D3MHub	319
8.3.1	struct D3MHub.ilk	319
8.3.2	modifier auth()	320
8.3.3	modifier lock()	320
8.3.4	constructor(daiJoin_) X	320
8.3.5	_min(x, y)	320
8.3.6	_max(x, y)	320
8.3.7	_divup(x, y)	320
8.3.8	rely(usr) X a	321
8.3.9	deny(usr) X a	321
8.3.10	file(what, data) X a	321
8.3.11	file(ilk, what, data) X a	321
8.3.12	file(ilk, what, data) X a	321
8.3.13	_wipe(ilk, _pool, urn)	322
8.3.14	_exec(ilk, _pool, Art, lineWad)	322
8.3.15	exec(ilk) X	324
8.3.16	exit(ilk, usr, wad) X	325
8.3.17	cage(ilk) X a	325
8.3.18	cull(ilk) X	326
8.3.19	uncull(ilk) X	326
8.3.20	pool(ilk)	327
8.3.21	plan(ilk)	327
8.3.22	tau(ilk)	327
8.3.23	culled(ilk)	327
8.3.24	tic(ilk)	327
8.4	contract D3MCoreDeployScript	329
8.4.1	struct StdChains.ChainData	331
8.4.2	struct StdChains.Chain	331
8.4.3	struct StdCheatsSafe.RawTx1559	331
8.4.4	struct StdCheatsSafe.RawTx1559Detail	331
8.4.5	struct StdCheatsSafe.Tx1559	332
8.4.6	struct StdCheatsSafe.Tx1559Detail	332
8.4.7	struct StdCheatsSafe.TxLegacy	332
8.4.8	struct StdCheatsSafe.TxDetailLegacy	332
8.4.9	struct StdCheatsSafe.AccessList	333
8.4.10	struct StdCheatsSafe.RawReceipt	333
8.4.11	struct StdCheatsSafe.Receipt	333
8.4.12	struct StdCheatsSafe.EIP1559ScriptArtifact	333
8.4.13	struct StdCheatsSafe.RawEIP1559ScriptArtifact	334
8.4.14	struct StdCheatsSafe.RawReceiptLog	334
8.4.15	struct StdCheatsSafe.ReceiptLog	334
8.4.16	struct StdCheatsSafe.TxReturn	334
8.4.17	modifier skipWhenForking() [StdCheatsSafe]	334
8.4.18	modifier skipWhenNotForking() [StdCheatsSafe]	335
8.4.19	modifier noGasMetering() [StdCheatsSafe]	335
8.4.20	_bound(x, min, max) [StdUtils]	335
8.4.21	bound(x, min, max) [StdUtils]	336
8.4.22	bound(x, min, max) [StdUtils]	336

8.4.23	computeCreateAddress(deployer, nonce) [StdUtils]	336
8.4.24	computeCreate2Address(salt, initcodeHash, deployer) [StdUtils]	337
8.4.25	bytesToUint(b) [StdUtils]	337
8.4.26	addressFromLast20Bytes(bytesValue) [StdUtils]	337
8.4.27	console2_log(p0, p1) [StdUtils]	337
8.4.28	console2_log(p0, p1) [StdUtils]	338
8.4.29	assumeNoPrecompiles(addr) [StdCheatsSafe]	338
8.4.30	assumeNoPrecompiles(addr, chainId) [StdCheatsSafe]	338
8.4.31	readEIP1559ScriptArtifact(path) [StdCheatsSafe]	339
8.4.32	rawToConvertedEIP1559s(rawTxs) [StdCheatsSafe]	339
8.4.33	rawToConvertedEIP1559(rawTx) [StdCheatsSafe]	339
8.4.34	rawToConvertedEIP1559Detail(rawDetail) [StdCheatsSafe]	339
8.4.35	readTx1559s(path) [StdCheatsSafe]	340
8.4.36	readTx1559(path, index) [StdCheatsSafe]	340
8.4.37	readReceipts(path) [StdCheatsSafe]	340
8.4.38	readReceipt(path, index) [StdCheatsSafe]	340
8.4.39	rawToConvertedReceipts(rawReceipts) [StdCheatsSafe]	341
8.4.40	rawToConvertedReceipt(rawReceipt) [StdCheatsSafe]	341
8.4.41	rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe]	341
8.4.42	deployCode(what, args) [StdCheatsSafe]	341
8.4.43	deployCode(what) [StdCheatsSafe]	342
8.4.44	deployCode(what, args, val) [StdCheatsSafe]	342
8.4.45	deployCode(what, val) [StdCheatsSafe]	342
8.4.46	makeAddrAndKey(name) [StdCheatsSafe]	342
8.4.47	makeAddr(name) [StdCheatsSafe]	343
8.4.48	deriveRememberKey(mnemonic, index) [StdCheatsSafe]	343
8.4.49	_bytesToUint(b) [StdCheatsSafe]	343
8.4.50	isFork() [StdCheatsSafe]	343
8.4.51	getChain(chainAlias) [StdChains]	343
8.4.52	getChain(chainId) [StdChains]	343
8.4.53	setChain(chainAlias, chain) [StdChains]	344
8.4.54	setChain(chainAlias, chain) [StdChains]	344
8.4.55	_toUpper(str) [StdChains]	345
8.4.56	getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains]	345
8.4.57	initialize() [StdChains]	345
8.4.58	setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains]	346
8.4.59	run() X	346
8.5	contract D3MAavePool	349
8.5.1	modifier auth()	349
8.5.2	modifier onlyHub()	349
8.5.3	constructor(ilk_, hub_, dai_, pool_) X	349
8.5.4	_rdiv(x, y)	350
8.5.5	_min(x, y)	350
8.5.6	rely(usr) X a	350
8.5.7	deny(usr) X a	350
8.5.8	file(what, data) X a	350
8.5.9	deposit(wad) X	350
8.5.10	withdraw(wad) X	351
8.5.11	exit(dst, wad) X	351
8.5.12	quit(dst) X a	351
8.5.13	assetBalance()	351
8.5.14	maxDeposit()	351
8.5.15	maxWithdraw()	351
8.5.16	redeemable()	352
8.5.17	collect() X	352
8.6	contract D3MOracle	353
8.6.1	modifier auth()	353
8.6.2	constructor(vat_, ilk_) X	353
8.6.3	rely(usr) X a	353
8.6.4	deny(usr) X a	353
8.6.5	file(what, data) X a	354
8.6.6	peek()	354
8.6.7	read()	354

8.7	contract D3MMom	355
8.7.1	modifier onlyOwner()	355
8.7.2	modifier auth()	355
8.7.3	constructor () X	355
8.7.4	isAuthorized(src, sig)	355
8.7.5	setOwner(owner_) X	355
8.7.6	setAuthority(authority_) X	356
8.7.7	disable(who) X a	356
8.8	contract D3MCoreInitScript	357
8.8.1	struct StdChains.ChainData	359
8.8.2	struct StdChains.Chain	359
8.8.3	struct StdCheatsSafe.RawTx1559	359
8.8.4	struct StdCheatsSafe.RawTx1559Detail	359
8.8.5	struct StdCheatsSafe.Tx1559	360
8.8.6	struct StdCheatsSafe.Tx1559Detail	360
8.8.7	struct StdCheatsSafe.TxLegacy	360
8.8.8	struct StdCheatsSafe.TxDetailLegacy	360
8.8.9	struct StdCheatsSafe.AccessList	361
8.8.10	struct StdCheatsSafe.RawReceipt	361
8.8.11	struct StdCheatsSafe.Receipt	361
8.8.12	struct StdCheatsSafe.EIP1559ScriptArtifact	361
8.8.13	struct StdCheatsSafe.RawEIP1559ScriptArtifact	362
8.8.14	struct StdCheatsSafe.RawReceiptLog	362
8.8.15	struct StdCheatsSafe.ReceiptLog	362
8.8.16	struct StdCheatsSafe.TxReturn	362
8.8.17	modifier skipWhenForking() [StdCheatsSafe]	362
8.8.18	modifier skipWhenNotForking() [StdCheatsSafe]	363
8.8.19	modifier noGasMetering() [StdCheatsSafe]	363
8.8.20	_bound(x, min, max) [StdUtils]	363
8.8.21	bound(x, min, max) [StdUtils]	364
8.8.22	bound(x, min, max) [StdUtils]	364
8.8.23	computeCreateAddress(deployer, nonce) [StdUtils]	364
8.8.24	computeCreate2Address(salt, initcodeHash, deployer) [StdUtils]	365
8.8.25	bytesToUint(b) [StdUtils]	365
8.8.26	addressFromLast20Bytes(bytesValue) [StdUtils]	365
8.8.27	console2_log(p0, p1) [StdUtils]	365
8.8.28	console2_log(p0, p1) [StdUtils]	366
8.8.29	assumeNoPrecompiles(addr) [StdCheatsSafe]	366
8.8.30	assumeNoPrecompiles(addr, chainId) [StdCheatsSafe]	366
8.8.31	readEIP1559ScriptArtifact(path) [StdCheatsSafe]	367
8.8.32	rawToConvertedEIPTx1559s(rawTxs) [StdCheatsSafe]	367
8.8.33	rawToConvertedEIPTx1559(rawTx) [StdCheatsSafe]	367
8.8.34	rawToConvertedEIP1559Detail(rawDetail) [StdCheatsSafe]	367
8.8.35	readTx1559s(path) [StdCheatsSafe]	368
8.8.36	readTx1559(path, index) [StdCheatsSafe]	368
8.8.37	readReceipts(path) [StdCheatsSafe]	368
8.8.38	readReceipt(path, index) [StdCheatsSafe]	368
8.8.39	rawToConvertedReceipts(rawReceipts) [StdCheatsSafe]	369
8.8.40	rawToConvertedReceipt(rawReceipt) [StdCheatsSafe]	369
8.8.41	rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe]	369
8.8.42	deployCode(what, args) [StdCheatsSafe]	369
8.8.43	deployCode(what) [StdCheatsSafe]	370
8.8.44	deployCode(what, args, val) [StdCheatsSafe]	370
8.8.45	deployCode(what, val) [StdCheatsSafe]	370
8.8.46	makeAddrAndKey(name) [StdCheatsSafe]	370
8.8.47	makeAddr(name) [StdCheatsSafe]	371
8.8.48	deriveRememberKey(mnemonic, index) [StdCheatsSafe]	371
8.8.49	_bytesToUint(b) [StdCheatsSafe]	371
8.8.50	isFork() [StdCheatsSafe]	371
8.8.51	getChain(chainAlias) [StdChains]	371
8.8.52	getChain(chainId) [StdChains]	371
8.8.53	setChain(chainAlias, chain) [StdChains]	372
8.8.54	setChain(chainAlias, chain) [StdChains]	372

8.8.55	<code>_toUpper(str)</code> [StdChains]	373
8.8.56	<code>getChainWithUpdatedRpcUrl(chainAlias, chain)</code> [StdChains]	373
8.8.57	<code>initialize()</code> [StdChains]	373
8.8.58	<code>setChainWithDefaultRpcUrl(chainAlias, chain)</code> [StdChains]	374
8.8.59	<code>run()</code> X	374
8.9	contract <code>D3MCompoundPool</code>	377
8.9.1	modifier <code>auth()</code>	377
8.9.2	modifier <code>onlyHub()</code>	377
8.9.3	constructor (<code>ilk_</code> , <code>hub_</code> , <code>cDai_</code>) X	377
8.9.4	<code>_wmul(x, y)</code>	378
8.9.5	<code>_wdiv(x, y)</code>	378
8.9.6	<code>_min(x, y)</code>	378
8.9.7	<code>rely(usr)</code> X a	378
8.9.8	<code>deny(usr)</code> X a	378
8.9.9	<code>file(what, data)</code> X a	378
8.9.10	<code>deposit(wad)</code> X	378
8.9.11	<code>withdraw(wad)</code> X	379
8.9.12	<code>exit(dst, wad)</code> X	379
8.9.13	<code>quit(dst)</code> X a	379
8.9.14	<code>preDebtChange()</code> X	379
8.9.15	<code>assetBalance()</code>	379
8.9.16	<code>maxDeposit()</code>	379
8.9.17	<code>maxWithdraw()</code>	379
8.9.18	<code>redeemable()</code>	380
8.9.19	<code>collect(claim)</code> X	380
8.10	contract <code>OptionalLoadDependencies</code>	381
8.10.1	<code>loadDependencies(name)</code> X	381
8.11	contract <code>D3MDeployScript</code>	383
8.11.1	struct <code>StdChains.ChainData</code>	385
8.11.2	struct <code>StdChains.Chain</code>	385
8.11.3	struct <code>StdCheatsSafe.RawTx1559</code>	385
8.11.4	struct <code>StdCheatsSafe.RawTx1559Detail</code>	385
8.11.5	struct <code>StdCheatsSafe.Tx1559</code>	386
8.11.6	struct <code>StdCheatsSafe.Tx1559Detail</code>	386
8.11.7	struct <code>StdCheatsSafe.TxLegacy</code>	386
8.11.8	struct <code>StdCheatsSafe.TxDetailLegacy</code>	386
8.11.9	struct <code>StdCheatsSafe.AccessList</code>	387
8.11.10	struct <code>StdCheatsSafe.RawReceipt</code>	387
8.11.11	struct <code>StdCheatsSafe.Receipt</code>	387
8.11.12	struct <code>StdCheatsSafe.EIP1559ScriptArtifact</code>	387
8.11.13	struct <code>StdCheatsSafe.RawEIP1559ScriptArtifact</code>	388
8.11.14	struct <code>StdCheatsSafe.RawReceiptLog</code>	388
8.11.15	struct <code>StdCheatsSafe.ReceiptLog</code>	388
8.11.16	struct <code>StdCheatsSafe.TxReturn</code>	388
8.11.17	modifier <code>skipWhenForking()</code> [StdCheatsSafe]	388
8.11.18	modifier <code>skipWhenNotForking()</code> [StdCheatsSafe]	389
8.11.19	modifier <code>noGasMetering()</code> [StdCheatsSafe]	389
8.11.20	<code>_bound(x, min, max)</code> [StdUtils]	389
8.11.21	<code>bound(x, min, max)</code> [StdUtils]	390
8.11.22	<code>bound(x, min, max)</code> [StdUtils]	390
8.11.23	<code>computeCreateAddress(deployer, nonce)</code> [StdUtils]	390
8.11.24	<code>computeCreate2Address(salt, initcodeHash, deployer)</code> [StdUtils]	391
8.11.25	<code>bytesToUint(b)</code> [StdUtils]	391
8.11.26	<code>addressFromLast20Bytes(bytesValue)</code> [StdUtils]	391
8.11.27	<code>console2_log(p0, p1)</code> [StdUtils]	391
8.11.28	<code>console2_log(p0, p1)</code> [StdUtils]	392
8.11.29	<code>assumeNoPrecompiles(addr)</code> [StdCheatsSafe]	392
8.11.30	<code>assumeNoPrecompiles(addr, chainId)</code> [StdCheatsSafe]	392
8.11.31	<code>readEIP1559ScriptArtifact(path)</code> [StdCheatsSafe]	393
8.11.32	<code>rawToConvertedEIP1559s(rawTx)</code> [StdCheatsSafe]	393
8.11.33	<code>rawToConvertedEIP1559(rawTx)</code> [StdCheatsSafe]	393
8.11.34	<code>rawToConvertedEIP1559Detail(rawDetail)</code> [StdCheatsSafe]	393
8.11.35	<code>readTx1559s(path)</code> [StdCheatsSafe]	394

8.11.36	readTx1559(path, index) [StdCheatsSafe]	394
8.11.37	readReceipts(path) [StdCheatsSafe]	394
8.11.38	readReceipt(path, index) [StdCheatsSafe]	394
8.11.39	rawToConvertedReceipts(rawReceipts) [StdCheatsSafe]	395
8.11.40	rawToConvertedReceipt(rawReceipt) [StdCheatsSafe]	395
8.11.41	rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe]	395
8.11.42	deployCode(what, args) [StdCheatsSafe]	395
8.11.43	deployCode(what) [StdCheatsSafe]	396
8.11.44	deployCode(what, args, val) [StdCheatsSafe]	396
8.11.45	deployCode(what, val) [StdCheatsSafe]	396
8.11.46	makeAddrAndKey(name) [StdCheatsSafe]	396
8.11.47	makeAddr(name) [StdCheatsSafe]	397
8.11.48	deriveRememberKey(mnemonic, index) [StdCheatsSafe]	397
8.11.49	_bytesToUint(b) [StdCheatsSafe]	397
8.11.50	isFork() [StdCheatsSafe]	397
8.11.51	getChain(chainAlias) [StdChains]	397
8.11.52	getChain(chainId) [StdChains]	397
8.11.53	setChain(chainAlias, chain) [StdChains]	398
8.11.54	setChain(chainAlias, chain) [StdChains]	398
8.11.55	_toUpper(str) [StdChains]	399
8.11.56	getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains]	399
8.11.57	initialize() [StdChains]	399
8.11.58	setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains]	400
8.11.59	run() X	400
8.12	contract D3MInitScript	403
8.12.1	struct StdChains.ChainData	405
8.12.2	struct StdChains.Chain	405
8.12.3	struct StdCheatsSafe.RawTx1559	405
8.12.4	struct StdCheatsSafe.RawTx1559Detail	405
8.12.5	struct StdCheatsSafe.Tx1559	406
8.12.6	struct StdCheatsSafe.Tx1559Detail	406
8.12.7	struct StdCheatsSafe.TxLegacy	406
8.12.8	struct StdCheatsSafe.TxDetailLegacy	406
8.12.9	struct StdCheatsSafe.AccessList	407
8.12.10	struct StdCheatsSafe.RawReceipt	407
8.12.11	struct StdCheatsSafe.Receipt	407
8.12.12	struct StdCheatsSafe.EIP1559ScriptArtifact	407
8.12.13	struct StdCheatsSafe.RawEIP1559ScriptArtifact	408
8.12.14	struct StdCheatsSafe.RawReceiptLog	408
8.12.15	struct StdCheatsSafe.ReceiptLog	408
8.12.16	struct StdCheatsSafe.TxReturn	408
8.12.17	modifier skipWhenForking() [StdCheatsSafe]	409
8.12.18	modifier skipWhenNotForking() [StdCheatsSafe]	409
8.12.19	modifier noGasMetering() [StdCheatsSafe]	409
8.12.20	_bound(x, min, max) [StdUtils]	409
8.12.21	bound(x, min, max) [StdUtils]	410
8.12.22	bound(x, min, max) [StdUtils]	410
8.12.23	computeCreateAddress(deployer, nonce) [StdUtils]	410
8.12.24	computeCreate2Address(salt, initcodeHash, deployer) [StdUtils]	411
8.12.25	bytesToUint(b) [StdUtils]	411
8.12.26	addressFromLast20Bytes(bytesValue) [StdUtils]	412
8.12.27	console2_log(p0, p1) [StdUtils]	412
8.12.28	console2_log(p0, p1) [StdUtils]	412
8.12.29	assumeNoPrecompiles(addr) [StdCheatsSafe]	412
8.12.30	assumeNoPrecompiles(addr, chainId) [StdCheatsSafe]	412
8.12.31	readEIP1559ScriptArtifact(path) [StdCheatsSafe]	413
8.12.32	rawToConvertedEIP1559s(rawTx) [StdCheatsSafe]	413
8.12.33	rawToConvertedEIP1559(rawTx) [StdCheatsSafe]	413
8.12.34	rawToConvertedEIP1559Detail(rawDetail) [StdCheatsSafe]	414
8.12.35	readTx1559s(path) [StdCheatsSafe]	414
8.12.36	readTx1559(path, index) [StdCheatsSafe]	414
8.12.37	readReceipts(path) [StdCheatsSafe]	414
8.12.38	readReceipt(path, index) [StdCheatsSafe]	414

8.12.39	rawToConvertedReceipts(rawReceipts) [StdCheatsSafe]	415
8.12.40	rawToConvertedReceipt(rawReceipt) [StdCheatsSafe]	415
8.12.41	rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe]	415
8.12.42	deployCode(what, args) [StdCheatsSafe]	416
8.12.43	deployCode(what) [StdCheatsSafe]	416
8.12.44	deployCode(what, args, val) [StdCheatsSafe]	416
8.12.45	deployCode(what, val) [StdCheatsSafe]	416
8.12.46	makeAddrAndKey(name) [StdCheatsSafe]	417
8.12.47	makeAddr(name) [StdCheatsSafe]	417
8.12.48	deriveRememberKey(mnemonic, index) [StdCheatsSafe]	417
8.12.49	_bytesToUint(b) [StdCheatsSafe]	417
8.12.50	isFork() [StdCheatsSafe]	417
8.12.51	getChain(chainAlias) [StdChains]	417
8.12.52	getChain(chainId) [StdChains]	418
8.12.53	setChain(chainAlias, chain) [StdChains]	418
8.12.54	setChain(chainAlias, chain) [StdChains]	419
8.12.55	_toUpper(str) [StdChains]	419
8.12.56	getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains]	419
8.12.57	initialize() [StdChains]	419
8.12.58	setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains]	420
8.12.59	run() X	420

9 Gem joins

423

9.1	contract GemJoin	423
9.1.1	modifier auth()	423
9.1.2	rely(usr) X a	424
9.1.3	deny(usr) X a	424
9.1.4	constructor(vat_, ilk_, gem_) X	424
9.1.5	cage() X a	424
9.1.6	join(usr, wad) X	424
9.1.7	exit(usr, wad) X	424
9.2	contract DaiJoin	425
9.2.1	modifier auth()	425
9.2.2	rely(usr) X a	425
9.2.3	deny(usr) X a	425
9.2.4	constructor(vat_, dai_) X	425
9.2.5	cage() X a	425
9.2.6	mul(x, y)	426
9.2.7	join(usr, wad) X	426
9.2.8	exit(usr, wad) X	426
9.3	contract GemJoin2	427
9.3.1	modifier auth()	427
9.3.2	rely(usr) X a	427
9.3.3	deny(usr) X a	427
9.3.4	constructor(vat_, ilk_, gem_) X	427
9.3.5	cage() X a	427
9.3.6	mul(x, y)	428
9.3.7	join(usr, wad) X	428
9.3.8	exit(usr, wad) X	428
9.4	contract GemJoin3	429
9.4.1	modifier auth()	429
9.4.2	rely(usr) X a	429
9.4.3	deny(usr) X a	429
9.4.4	constructor(vat_, ilk_, gem_, decimals) X	429
9.4.5	cage() X a	429
9.4.6	mul(x, y)	430
9.4.7	join(usr, amt) X	430
9.4.8	exit(usr, amt) X	430
9.5	contract GemBag	431
9.5.1	constructor(lad_, gem_) X	431
9.5.2	exit(usr, wad) X	431
9.6	contract GemJoin4	433
9.6.1	modifier auth()	433

9.6.2	rely(usr) X a	433
9.6.3	deny(usr) X a	433
9.6.4	constructor(vat_, ilk_, gem_) X	433
9.6.5	cage() X a	433
9.6.6	make() X	434
9.6.7	make(usr) X	434
9.6.8	join(usr, wad) X	434
9.6.9	exit(usr, wad) X	434
9.7	contract GemJoin5	435
9.7.1	modifier auth()	435
9.7.2	rely(usr) X a	435
9.7.3	deny(usr) X a	435
9.7.4	constructor(vat_, ilk_, gem_) X	435
9.7.5	cage() X a	435
9.7.6	mul(x, y)	436
9.7.7	join(usr, amt) X	436
9.7.8	exit(usr, amt) X	436
9.8	contract GemJoin6	437
9.8.1	modifier auth()	437
9.8.2	rely(usr) X a	437
9.8.3	deny(usr) X a	437
9.8.4	constructor(vat_, ilk_, gem_) X	437
9.8.5	cage() X a	438
9.8.6	setImplementation(implementation, permitted) X a	438
9.8.7	join(usr, wad) X	438
9.8.8	exit(usr, wad) X	438
9.9	contract GemJoin7	439
9.9.1	modifier auth()	439
9.9.2	rely(usr) X a	439
9.9.3	deny(usr) X a	439
9.9.4	constructor(vat_, ilk_, gem_) X	439
9.9.5	cage() X a	440
9.9.6	setImplementation(implementation, permitted) X a	440
9.9.7	mul(x, y)	440
9.9.8	sub(x, y)	440
9.9.9	join(usr, amt) X	440
9.9.10	exit(usr, amt) X	440
9.10	contract GemJoin8	441
9.10.1	modifier auth()	441
9.10.2	rely(usr) X a	441
9.10.3	deny(usr) X a	441
9.10.4	constructor(vat_, ilk_, gem_) X	441
9.10.5	cage() X a	442
9.10.6	setImplementation(implementation, permitted) X a	442
9.10.7	mul(x, y)	442
9.10.8	join(usr, amt) X	442
9.10.9	exit(usr, amt) X	442
9.11	contract GemJoin9	443
9.11.1	modifier auth()	443
9.11.2	constructor(vat_, ilk_, gem_) X	443
9.11.3	rely(usr) X a	443
9.11.4	deny(usr) X a	443
9.11.5	cage() X a	444
9.11.6	_add(x, y)	444
9.11.7	_sub(x, y)	444
9.11.8	join(usr) X	444
9.11.9	join(usr, wad) X	444
9.11.10	_join(usr)	444
9.11.11	exit(usr, wad) X	445
9.12	contract AuthGemJoin	447
9.12.1	modifier auth()	447
9.12.2	rely(usr) X a	447
9.12.3	deny(usr) X a	447

9.12.4	<code>constructor(vat_, ilk_, gem_) X</code>	447
9.12.5	<code>cage() X a</code>	447
9.12.6	<code>join(usr, wad) X a</code>	448
9.12.7	<code>exit(usr, wad) X</code>	448
9.13	<code>contract ManagedGemJoin</code>	449
9.13.1	<code>modifier auth()</code>	449
9.13.2	<code>rely(usr) X a</code>	449
9.13.3	<code>deny(usr) X a</code>	449
9.13.4	<code>constructor(vat_, ilk_, gem_) X</code>	449
9.13.5	<code>cage() X a</code>	450
9.13.6	<code>_mul(x, y)</code>	450
9.13.7	<code>join(usr, amt) X a</code>	450
9.13.8	<code>exit(urn, usr, amt) X a</code>	450
9.14	<code>contract AAVE</code>	451
9.14.1	<code>add(x, y)</code>	451
9.14.2	<code>sub(x, y)</code>	451
9.14.3	<code>constructor(supply) X</code>	451
9.14.4	<code>totalSupply()</code>	451
9.14.5	<code>balanceOf(src)</code>	451
9.14.6	<code>allowance(src, guy)</code>	451
9.14.7	<code>transfer(dst, wad) X</code>	451
9.14.8	<code>transferFrom(src, dst, wad) X</code>	452
9.14.9	<code>approve(guy, wad) X</code>	452
9.15	<code>contract BAL</code>	453
9.15.1	<code>add(x, y)</code>	453
9.15.2	<code>sub(x, y)</code>	453
9.15.3	<code>constructor(supply) X</code>	453
9.15.4	<code>totalSupply()</code>	453
9.15.5	<code>balanceOf(src)</code>	453
9.15.6	<code>allowance(src, guy)</code>	453
9.15.7	<code>transfer(dst, wad) X</code>	453
9.15.8	<code>transferFrom(src, dst, wad) X</code>	454
9.15.9	<code>approve(guy, wad) X</code>	454
9.16	<code>contract BAT</code>	455
9.16.1	<code>add(x, y)</code>	455
9.16.2	<code>sub(x, y)</code>	455
9.16.3	<code>constructor(supply) X</code>	455
9.16.4	<code>totalSupply()</code>	455
9.16.5	<code>balanceOf(src)</code>	455
9.16.6	<code>allowance(src, guy)</code>	455
9.16.7	<code>transfer(dst, wad) X</code>	455
9.16.8	<code>transferFrom(src, dst, wad) X</code>	456
9.16.9	<code>approve(guy, wad) X</code>	456
9.17	<code>contract COMP</code>	457
9.17.1	<code>add(x, y)</code>	457
9.17.2	<code>sub(x, y)</code>	457
9.17.3	<code>constructor(supply) X</code>	457
9.17.4	<code>totalSupply()</code>	457
9.17.5	<code>balanceOf(src)</code>	457
9.17.6	<code>allowance(src, guy)</code>	457
9.17.7	<code>transfer(dst, wad) X</code>	457
9.17.8	<code>transferFrom(src, dst, wad) X</code>	458
9.17.9	<code>approve(guy, wad) X</code>	458
9.18	<code>contract DGD</code>	459
9.18.1	<code>add(x, y)</code>	459
9.18.2	<code>sub(x, y)</code>	459
9.18.3	<code>constructor(supply) X</code>	459
9.18.4	<code>totalSupply()</code>	459
9.18.5	<code>balanceOf(src)</code>	459
9.18.6	<code>allowance(src, guy)</code>	459
9.18.7	<code>transfer(dst, wad) X</code>	459
9.18.8	<code>transferFrom(src, dst, wad) X</code>	460
9.18.9	<code>approve(guy, wad) X</code>	460

9.19	<code>contract</code> GNT	461
9.19.1	<code>add(x, y)</code>	461
9.19.2	<code>sub(x, y)</code>	461
9.19.3	<code>constructor(supply)</code> X	461
9.19.4	<code>totalSupply()</code>	461
9.19.5	<code>balanceOf(src)</code>	461
9.19.6	<code>transfer(dst, wad)</code> X	461
9.20	<code>contract</code> GUSD	463
9.20.1	<code>add(x, y)</code>	463
9.20.2	<code>sub(x, y)</code>	463
9.20.3	<code>constructor(supply)</code> X	463
9.20.4	<code>setImplementation(newImplementation)</code> X	463
9.20.5	<code>totalSupply()</code>	463
9.20.6	<code>balanceOf(src)</code>	463
9.20.7	<code>allowance(src, guy)</code>	463
9.20.8	<code>transfer(dst, wad)</code> X	464
9.20.9	<code>transferFrom(src, dst, wad)</code> X	464
9.20.10	<code>approve(guy, wad)</code> X	464
9.21	<code>contract</code> KNC	465
9.21.1	<code>add(x, y)</code>	465
9.21.2	<code>sub(x, y)</code>	465
9.21.3	<code>constructor(supply)</code> X	465
9.21.4	<code>totalSupply()</code>	465
9.21.5	<code>balanceOf(src)</code>	465
9.21.6	<code>allowance(src, guy)</code>	465
9.21.7	<code>transfer(dst, wad)</code> X	465
9.21.8	<code>transferFrom(src, dst, wad)</code> X	466
9.21.9	<code>approve(guy, wad)</code> X	466
9.22	<code>contract</code> LINK	467
9.22.1	<code>add(x, y)</code>	467
9.22.2	<code>sub(x, y)</code>	467
9.22.3	<code>constructor(supply)</code> X	467
9.22.4	<code>totalSupply()</code>	467
9.22.5	<code>balanceOf(src)</code>	467
9.22.6	<code>allowance(src, guy)</code>	467
9.22.7	<code>transfer(dst, wad)</code> X	467
9.22.8	<code>transferFrom(src, dst, wad)</code> X	468
9.22.9	<code>approve(guy, wad)</code> X	468
9.23	<code>contract</code> LRC	469
9.23.1	<code>add(x, y)</code>	469
9.23.2	<code>sub(x, y)</code>	469
9.23.3	<code>constructor(supply)</code> X	469
9.23.4	<code>totalSupply()</code>	469
9.23.5	<code>balanceOf(src)</code>	469
9.23.6	<code>allowance(src, guy)</code>	469
9.23.7	<code>transfer(dst, wad)</code> X	469
9.23.8	<code>transferFrom(src, dst, wad)</code> X	470
9.23.9	<code>approve(guy, wad)</code> X	470
9.24	<code>contract</code> MANA	471
9.24.1	<code>add(x, y)</code>	471
9.24.2	<code>sub(x, y)</code>	471
9.24.3	<code>constructor(supply)</code> X	471
9.24.4	<code>totalSupply()</code>	471
9.24.5	<code>balanceOf(src)</code>	471
9.24.6	<code>allowance(src, guy)</code>	471
9.24.7	<code>transfer(dst, wad)</code> X	471
9.24.8	<code>transferFrom(src, dst, wad)</code> X	472
9.24.9	<code>approve(guy, wad)</code> X	472
9.25	<code>contract</code> MATIC	473
9.25.1	<code>add(x, y)</code>	473
9.25.2	<code>sub(x, y)</code>	473
9.25.3	<code>constructor(supply)</code> X	473
9.25.4	<code>totalSupply()</code>	473

9.25.5	balanceOf(src)	473
9.25.6	allowance(src, guy)	473
9.25.7	transfer(dst, wad) X	473
9.25.8	transferFrom(src, dst, wad) X	474
9.25.9	approve(guy, wad) X	474
9.26	contract OMG	475
9.26.1	add(x, y)	475
9.26.2	sub(x, y)	475
9.26.3	constructor(supply) X	475
9.26.4	totalSupply()	475
9.26.5	balanceOf(src)	475
9.26.6	allowance(src, guy)	475
9.26.7	transfer(dst, wad) X	475
9.26.8	transferFrom(src, dst, wad) X	476
9.26.9	approve(guy, wad) X	476
9.27	contract PAXG	477
9.27.1	add(x, y)	477
9.27.2	sub(x, y)	477
9.27.3	mul(a, b)	477
9.27.4	div(a, b)	477
9.27.5	constructor(supply, fee) X	478
9.27.6	totalSupply()	478
9.27.7	balanceOf(src)	478
9.27.8	allowance(src, guy)	478
9.27.9	transfer(dst, wad) X	478
9.27.10	transferFrom(src, dst, wad) X	478
9.27.11	approve(guy, wad) X	479
9.27.12	setFeeRate(_newFeeRate)	479
9.27.13	getFeeFor(_value)	479
9.28	contract PAXUSD	481
9.28.1	add(x, y)	481
9.28.2	sub(x, y)	481
9.28.3	constructor(supply) X	481
9.28.4	totalSupply()	481
9.28.5	balanceOf(src)	481
9.28.6	allowance(src, guy)	481
9.28.7	transfer(dst, wad) X	481
9.28.8	transferFrom(src, dst, wad) X	482
9.28.9	approve(guy, wad) X	482
9.29	contract RENBTC	483
9.29.1	add(x, y)	483
9.29.2	sub(x, y)	483
9.29.3	constructor(supply) X	483
9.29.4	totalSupply()	483
9.29.5	balanceOf(src)	483
9.29.6	allowance(src, guy)	483
9.29.7	transfer(dst, wad) X	483
9.29.8	transferFrom(src, dst, wad) X	484
9.29.9	approve(guy, wad) X	484
9.30	contract REP	485
9.30.1	add(x, y)	485
9.30.2	sub(x, y)	485
9.30.3	constructor(supply) X	485
9.30.4	totalSupply()	485
9.30.5	balanceOf(src)	485
9.30.6	allowance(src, guy)	485
9.30.7	transfer(dst, wad) X	485
9.30.8	transferFrom(src, dst, wad) X	486
9.30.9	approve(guy, wad) X	486
9.31	contract TUSD	487
9.31.1	add(x, y)	487
9.31.2	sub(x, y)	487
9.31.3	constructor(supply) X	487

9.31.4	setImplementation(newImplementation) X	487
9.31.5	totalSupply()	487
9.31.6	balanceOf(src)	487
9.31.7	allowance(src, guy)	487
9.31.8	transfer(dst, wad) X	488
9.31.9	transferFrom(src, dst, wad) X	488
9.31.10	approve(guy, wad) X	488
9.32	contract UNI	489
9.32.1	add(x, y)	489
9.32.2	sub(x, y)	489
9.32.3	constructor(supply) X	489
9.32.4	totalSupply()	489
9.32.5	balanceOf(src)	489
9.32.6	allowance(src, guy)	489
9.32.7	transfer(dst, wad) X	489
9.32.8	transferFrom(src, dst, wad) X	490
9.32.9	approve(guy, wad) X	490
9.33	contract USDC	491
9.33.1	add(x, y)	491
9.33.2	sub(x, y)	491
9.33.3	constructor(supply) X	491
9.33.4	totalSupply()	491
9.33.5	balanceOf(src)	491
9.33.6	allowance(src, guy)	491
9.33.7	transfer(dst, wad) X	491
9.33.8	transferFrom(src, dst, wad) X	492
9.33.9	approve(guy, wad) X	492
9.34	contract USDT	493
9.34.1	modifier onlyPayloadSize(size)	493
9.34.2	constructor(_initialSupply) X	493
9.34.3	changeFees(_basisPointsRate, _maximumFee) X	493
9.34.4	balanceOf(_owner)	493
9.34.5	transfer(_to, _value) X	493
9.34.6	transferFrom(_from, _to, _value) X	494
9.34.7	approve(_spender, _value) X	494
9.34.8	allowance(_owner, _spender)	494
9.34.9	deprecate(_upgradedAddress) X	494
9.35	contract WBTC	495
9.35.1	add(x, y)	495
9.35.2	sub(x, y)	495
9.35.3	constructor(supply) X	495
9.35.4	totalSupply()	495
9.35.5	balanceOf(src)	495
9.35.6	allowance(src, guy)	495
9.35.7	transfer(dst, wad) X	495
9.35.8	transferFrom(src, dst, wad) X	496
9.35.9	approve(guy, wad) X	496
9.36	contract WSTETH	497
9.36.1	add(x, y)	497
9.36.2	sub(x, y)	497
9.36.3	constructor(supply) X	497
9.36.4	totalSupply()	497
9.36.5	balanceOf(src)	497
9.36.6	allowance(src, guy)	497
9.36.7	transfer(dst, wad) X	497
9.36.8	transferFrom(src, dst, wad) X	498
9.36.9	approve(guy, wad) X	498
9.37	contract YFI	499
9.37.1	add(x, y)	499
9.37.2	sub(x, y)	499
9.37.3	constructor(supply) X	499
9.37.4	totalSupply()	499
9.37.5	balanceOf(src)	499

9.37.6	allowance(src, guy)	499
9.37.7	transfer(dst, wad) X	499
9.37.8	transferFrom(src, dst, wad) X	500
9.37.9	approve(guy, wad) X	500
9.38	contract ZRX	501
9.38.1	add(x, y)	501
9.38.2	sub(x, y)	501
9.38.3	constructor(supply) X	501
9.38.4	totalSupply()	501
9.38.5	balanceOf(src)	501
9.38.6	allowance(src, guy)	501
9.38.7	transfer(dst, wad) X	501
9.38.8	transferFrom(src, dst, wad) X	502
9.38.9	approve(guy, wad) X	502
10	Peg Stability	503
10.1	contract AuthGemJoin5	503
10.1.1	modifier auth()	503
10.1.2	rely(usr) X a	503
10.1.3	deny(usr) X a	503
10.1.4	constructor(vat_, ilk_, gem_) X	503
10.1.5	cage() X a	504
10.1.6	mul(x, y)	504
10.1.7	join(urn, amt, msgSender) X a	504
10.1.8	exit(usr, amt) X	504
10.2	contract AuthGemJoin8	505
10.2.1	modifier auth()	505
10.2.2	rely(usr) X a	505
10.2.3	deny(usr) X a	505
10.2.4	constructor(vat_, ilk_, gem_) X	505
10.2.5	cage() X a	506
10.2.6	setImplementation(implementation, permitted) X a	506
10.2.7	mul(x, y)	506
10.2.8	join(urn, amt, msgSender) X a	506
10.2.9	exit(usr, amt) X	506
10.3	contract AuthGemJoin(2)	507
10.3.1	modifier auth()	507
10.3.2	rely(usr) X a	507
10.3.3	deny(usr) X a	507
10.3.4	constructor(vat_, ilk_, gem_) X	507
10.3.5	cage() X a	507
10.3.6	join(urn, wad, msgSender) X a	508
10.3.7	exit(guy, wad) X	508
10.4	contract DssPsm	509
10.4.1	modifier auth()	509
10.4.2	rely(usr) X a	509
10.4.3	deny(usr) X a	509
10.4.4	constructor(gemJoin_, daiJoin_, vow_) X	509
10.4.5	add(x, y)	510
10.4.6	sub(x, y)	510
10.4.7	mul(x, y)	510
10.4.8	file(what, data) X a	510
10.4.9	hope(usr) X a	510
10.4.10	nope(usr) X a	510
10.4.11	sellGem(usr, gemAmt) X	510
10.4.12	buyGem(usr, gemAmt) X	511
11	WETH	513
11.1	contract DSWethFactory	513
11.1.1	make() X	513
11.2	contract WETH	515
11.3	contract WETH9	517
11.3.1	fallback() X	517

11.3.2	deposit() X	517
11.3.3	withdraw(wad) X	517
11.3.4	totalSupply()	517
11.3.5	approve(guy, wad) X	517
11.3.6	transfer(dst, wad) X	517
11.3.7	transferFrom(src, dst, wad) X	518
12	Proxy	519
12.1	contract UrnHandler	519
12.1.1	constructor(vat) X	519
12.2	contract DssCdpManager	521
12.2.1	struct DssCdpManager.List	521
12.2.2	modifier cdpAllowed(cdp)	521
12.2.3	modifier urnAllowed(urn)	521
12.2.4	constructor(vat_) X	522
12.2.5	add(x, y)	522
12.2.6	sub(x, y)	522
12.2.7	toInt(x)	522
12.2.8	cdpAllow(cdp, usr, ok) X	522
12.2.9	urnAllow(usr, ok) X	522
12.2.10	open(ilk, usr) X	522
12.2.11	give(cdp, dst) X	523
12.2.12	frob(cdp, dink, dart) X	524
12.2.13	flux(cdp, dst, wad) X	524
12.2.14	flux(ilk, cdp, dst, wad) X	524
12.2.15	move(cdp, dst, rad) X	524
12.2.16	quit(cdp, dst) X	524
12.2.17	enter(src, cdp) X	525
12.2.18	shift(cdpSrc, cdpDst) X	525
12.3	contract GetCdps	527
12.3.1	getCdpsAsc(manager, guy)	527
12.3.2	getCdpsDesc(manager, guy)	527
12.4	contract DssProxyActions	529
12.4.1	mul(x, y) [Common]	529
12.4.2	daiJoin_join(apt, urn, wad) [Common] X	529
12.4.3	sub(x, y)	529
12.4.4	toInt(x)	529
12.4.5	toRad(wad)	529
12.4.6	convertTo18(gemJoin, amt)	530
12.4.7	_getDrawDart(vat, jug, urn, ilk, wad)	530
12.4.8	_getWipeDart(vat, dai, urn, ilk)	530
12.4.9	_getWipeAllWad(vat, usr, urn, ilk)	530
12.4.10	transfer(gem, dst, amt) X	531
12.4.11	ethJoin_join(apt, urn) X	531
12.4.12	gemJoin_join(apt, urn, amt, transferFrom) X	531
12.4.13	hope(obj, usr) X	531
12.4.14	nope(obj, usr) X	532
12.4.15	open(manager, ilk, usr) X	532
12.4.16	give(manager, cdp, usr) X	532
12.4.17	giveToProxy(proxyRegistry, manager, cdp, dst) X	532
12.4.18	cdpAllow(manager, cdp, usr, ok) X	532
12.4.19	urnAllow(manager, usr, ok) X	533
12.4.20	flux(manager, cdp, dst, wad) X	533
12.4.21	move(manager, cdp, dst, rad) X	533
12.4.22	frob(manager, cdp, dink, dart) X	533
12.4.23	quit(manager, cdp, dst) X	533
12.4.24	enter(manager, src, cdp) X	534
12.4.25	shift(manager, cdpSrc, cdpOrg) X	534
12.4.26	makeGemBag(gemJoin) X	534
12.4.27	lockETH(manager, ethJoin, cdp) X	534
12.4.28	safeLockETH(manager, ethJoin, cdp, owner) X	534
12.4.29	lockGem(manager, gemJoin, cdp, amt, transferFrom) X	535
12.4.30	safeLockGem(manager, gemJoin, cdp, amt, transferFrom, owner) X	535

12.4.31	freeETH(manager, ethJoin, cdp, wad)	X	535
12.4.32	freeGem(manager, gemJoin, cdp, amt)	X	535
12.4.33	exitETH(manager, ethJoin, cdp, wad)	X	536
12.4.34	exitGem(manager, gemJoin, cdp, amt)	X	536
12.4.35	draw(manager, jug, daiJoin, cdp, wad)	X	536
12.4.36	wipe(manager, daiJoin, cdp, wad)	X	537
12.4.37	safeWipe(manager, daiJoin, cdp, wad, owner)	X	537
12.4.38	wipeAll(manager, daiJoin, cdp)	X	537
12.4.39	safeWipeAll(manager, daiJoin, cdp, owner)	X	538
12.4.40	lockETHAndDraw(manager, jug, ethJoin, daiJoin, cdp, wadD)	X	538
12.4.41	openLockETHAndDraw(manager, jug, ethJoin, daiJoin, ilk, wadD)	X	538
12.4.42	lockGemAndDraw(manager, jug, gemJoin, daiJoin, cdp, amtC, wadD, transferFrom)	X	539
12.4.43	openLockGemAndDraw(manager, jug, gemJoin, daiJoin, ilk, amtC, wadD, transferFrom)	X	539
12.4.44	openLockGNTAndDraw(manager, jug, gntJoin, daiJoin, ilk, amtC, wadD)	X	539
12.4.45	wipeAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC, wadD)	X	540
12.4.46	wipeAllAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC)	X	540
12.4.47	wipeAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC, wadD)	X	541
12.4.48	wipeAllAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC)	X	541
12.5	contract DssProxyActionsEnd		543
12.5.1	mul(x, y) [Common]		543
12.5.2	daiJoin_join(apt, urn, wad) [Common]	X	543
12.5.3	_free(manager, end, cdp)		543
12.5.4	freeETH(manager, ethJoin, end, cdp)	X	544
12.5.5	freeGem(manager, gemJoin, end, cdp)	X	544
12.5.6	pack(daiJoin, end, wad)	X	544
12.5.7	cashETH(ethJoin, end, ilk, wad)	X	544
12.5.8	cashGem(gemJoin, end, ilk, wad)	X	545
12.6	contract DssProxyActionsDsr		547
12.6.1	mul(x, y) [Common]		547
12.6.2	daiJoin_join(apt, urn, wad) [Common]	X	547
12.6.3	join(daiJoin, pot, wad)	X	547
12.6.4	exit(daiJoin, pot, wad)	X	548
12.6.5	exitAll(daiJoin, pot)	X	548
13	Exchange callees		549
13.1	contract CalleeMakerOtcDai		549
13.1.1	add(x, y) [CalleeMakerOtc]		549
13.1.2	sub(x, y) [CalleeMakerOtc]		549
13.1.3	divup(x, y) [CalleeMakerOtc]		549
13.1.4	setUp(otc_, daiJoin_) [CalleeMakerOtc]		549
13.1.5	_fromWad(gemJoin, wad) [CalleeMakerOtc]		550
13.1.6	constructor(otc_, daiJoin_)	X	550
13.1.7	clipperCall(sender, daiAmt, gemAmt, data)	X	550
13.2	contract CurveLpTokenUniv3Callee		551
13.2.1	struct CurveLpTokenUniv3Callee.CurveData		551
13.2.2	_add(x, y)		551
13.2.3	_sub(x, y)		551
13.2.4	_divup(x, y)		551
13.2.5	constructor(uniV3Router_, daiJoin_, weth_)	X	551
13.2.6	_fromWad(gemJoin, wad)		552
13.2.7	clipperCall(sender, owe, slice, data)	X	552
13.3	contract PSMCallee		555
13.3.1	_add(x, y)		555
13.3.2	_sub(x, y)		555
13.3.3	_divup(x, y)		555
13.3.4	constructor(daiJoin_)	X	555
13.3.5	_fromWad(gemJoin, wad)		555
13.3.6	clipperCall(sender, owe, slice, data)	X	555
13.4	contract UniswapV2CalleeDai		557
13.4.1	add(x, y) [UniswapV2Callee]		557
13.4.2	sub(x, y) [UniswapV2Callee]		557

13.4.3	divup(x, y) [UniswapV2Callee]	557
13.4.4	setUp(uniRouter02_, daiJoin_) [UniswapV2Callee]	557
13.4.5	_fromWad(gemJoin, wad) [UniswapV2Callee]	557
13.4.6	constructor(uniRouter02_, daiJoin_) X	557
13.4.7	clipperCall(sender, daiAmt, gemAmt, data) X	558
13.5	contract UniswapV2LpTokenCalleeDai	559
13.5.1	add(x, y) [UniswapV2Callee(2)]	559
13.5.2	sub(x, y) [UniswapV2Callee(2)]	559
13.5.3	divup(x, y) [UniswapV2Callee(2)]	559
13.5.4	setUp(uniRouter02_, daiJoin_) [UniswapV2Callee(2)]	559
13.5.5	_fromWad(gemJoin, wad) [UniswapV2Callee(2)]	559
13.5.6	constructor(uniRouter02_, daiJoin_) X	559
13.5.7	swapGemForDai(token, path, to)	560
13.5.8	clipperCall(sender, daiAmt, gemAmt, data) X	560
13.6	contract WstETHCurveUniv3Callee	563
13.6.1	_add(x, y)	563
13.6.2	_sub(x, y)	563
13.6.3	_divup(x, y)	563
13.6.4	constructor(curvePool_, uniV3Router_, daiJoin_, weth_) X	563
13.6.5	_fromWad(gemJoin, wad)	563
13.6.6	clipperCall(sender, owe, slice, data) X	564
13.7	contract UniswapV3Callee	567
13.7.1	_add(x, y)	567
13.7.2	_sub(x, y)	567
13.7.3	_divup(x, y)	567
13.7.4	constructor(uniV3Router_, daiJoin_) X	567
13.7.5	_fromWad(gemJoin, wad)	567
13.7.6	clipperCall(sender, owe, slice, data) X	567
14	Arbitrum bridge	569
14.1	contract L1DaiGateway	569
14.1.1	modifier onlyL2Counterpart(l2Counterpart) [L1CrossDomainEnabled]	569
14.1.2	modifier auth()	569
14.1.3	constructor(_inbox) [L1CrossDomainEnabled] X	570
14.1.4	sendTxToL2(target, user, maxSubmissionCost, maxGas, gasPriceBid, data) [L1CrossDomainEnabled]	570
14.1.5	sendTxToL2NoAliasing(target, user, l1CallValue, maxSubmissionCost, maxGas, gasPriceBid, data) [L1CrossDomainEnabled]	570
14.1.6	rely(usr) X a	570
14.1.7	deny(usr) X a	571
14.1.8	constructor(_l2Counterpart, _l1Router, _inbox, _l1Dai, _l2Dai, _l1Escrow) X	571
14.1.9	close() X a	571
14.1.10	outboundTransfer(l1Token, to, amount, maxGas, gasPriceBid, data) X	571
14.1.11	getOutboundCalldata(l1Token, from, to, amount, data)	572
14.1.12	finalizeInboundTransfer(l1Token, from, to, amount, data) X	572
14.1.13	parseOutboundData(data)	572
14.1.14	calculateL2TokenAddress(l1Token)	573
14.1.15	counterpartGateway()	573
14.2	contract L1Escrow	575
14.2.1	modifier auth()	575
14.2.2	rely(usr) X a	575
14.2.3	deny(usr) X a	575
14.2.4	constructor() X	575
14.2.5	approve(token, spender, value) X a	575
14.3	contract L1GovernanceRelay	577
14.3.1	modifier onlyL2Counterpart(l2Counterpart) [L1CrossDomainEnabled]	577
14.3.2	modifier auth()	577
14.3.3	constructor(_inbox) [L1CrossDomainEnabled] X	577
14.3.4	sendTxToL2(target, user, maxSubmissionCost, maxGas, gasPriceBid, data) [L1CrossDomainEnabled]	577
14.3.5	sendTxToL2NoAliasing(target, user, l1CallValue, maxSubmissionCost, maxGas, gasPriceBid, data) [L1CrossDomainEnabled]	578

14.3.6	rely(usr) X a	578
14.3.7	deny(usr) X a	578
14.3.8	constructor(_inbox, _l2GovernanceRelay) X	579
14.3.9	reclaim(receiver, amount) X a	579
14.3.10	relay(target, targetData, l1CallValue, maxGas, gasPriceBid, maxSubmissionCost) X a	579
14.4	contract L2DaiGateway	581
14.4.1	modifier onlyL1Counterpart(l1Counterpart) [L2CrossDomainEnabled] . .	581
14.4.2	modifier auth()	581
14.4.3	sendTxToL1(user, to, data) [L2CrossDomainEnabled]	581
14.4.4	applyL1ToL2Alias(l1Address) [L2CrossDomainEnabled]	581
14.4.5	rely(usr) X a	582
14.4.6	deny(usr) X a	582
14.4.7	constructor(_l1Counterpart, _l2Router, _l1Dai, _l2Dai) X	582
14.4.8	close() X a	582
14.4.9	outboundTransfer(l1Token, to, amount, data) X	582
14.4.10	outboundTransfer(l1Token, to, amount, , , data) X	583
14.4.11	getOutboundCalldata(token, from, to, amount, data)	583
14.4.12	finalizeInboundTransfer(l1Token, from, to, amount,) X	583
14.4.13	calculateL2TokenAddress(l1Token)	584
14.4.14	parseOutboundData(data)	584
14.4.15	counterpartGateway()	584
14.5	contract L2GovernanceRelay	585
14.5.1	modifier onlyL1Counterpart(l1Counterpart) [L2CrossDomainEnabled] . .	585
14.5.2	sendTxToL1(user, to, data) [L2CrossDomainEnabled]	585
14.5.3	applyL1ToL2Alias(l1Address) [L2CrossDomainEnabled]	585
14.5.4	constructor(_l1GovernanceRelay) X	585
14.5.5	relay(target, targetData) X	586
14.6	contract Dai(2)	587
14.6.1	modifier auth()	587
14.6.2	rely(usr) X a	587
14.6.3	deny(usr) X a	587
14.6.4	_add(x, y)	587
14.6.5	_sub(x, y)	588
14.6.6	constructor() X	588
14.6.7	_calculateDomainSeparator(chainId)	588
14.6.8	DOMAIN_SEPARATOR()	588
14.6.9	transfer(to, value) X	588
14.6.10	transferFrom(from, to, value) X	589
14.6.11	approve(spender, value) X	589
14.6.12	increaseAllowance(spender, addedValue) X	589
14.6.13	decreaseAllowance(spender, subtractedValue) X	589
14.6.14	mint(to, value) X a	590
14.6.15	burn(from, value) X	590
14.6.16	permit(owner, spender, value, deadline, v, r, s) X	590
15	Optimism bridge	591
15.1	contract L1DAITokenBridge	591
15.1.1	modifier onlyFromCrossDomainAccount(_sourceDomainAccount) [OVM_CrossDomainEnabled]	592
15.1.2	modifier auth()	592
15.1.3	constructor(_messenger) [OVM_CrossDomainEnabled] X	592
15.1.4	getCrossDomainMessenger() [OVM_CrossDomainEnabled]	592
15.1.5	sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message) [OVM_CrossDomainEnabled]	593
15.1.6	rely(usr) X a	593
15.1.7	deny(usr) X a	593
15.1.8	constructor(_l1Token, _l2DAITokenBridge, _l2Token, _l1messenger, _escrow) X	593
15.1.9	close() X a	594
15.1.10	depositERC20(_l1Token, _l2Token, _amount, _l2Gas, _data) X	594
15.1.11	depositERC20To(_l1Token, _l2Token, _to, _amount, _l2Gas, _data) X	594
15.1.12	_initiateERC20Deposit(_from, _to, _amount, _l2Gas, _data)	594

15.1.13	finalizeERC20Withdrawal(_l1Token, _l2Token, _from, _to, _amount, _data) X	595
15.2	contract L1Escrow(2)	597
15.2.1	modifier auth()	597
15.2.2	rely(usr) X a	597
15.2.3	deny(usr) X a	597
15.2.4	constructor() X	597
15.2.5	approve(token, spender, value) X a	597
15.3	contract L1GovernanceRelay(2)	599
15.3.1	modifier onlyFromCrossDomainAccount(_sourceDomainAccount) [OVM_CrossDomainEnabled]	599
15.3.2	modifier auth()	600
15.3.3	constructor(_messenger) [OVM_CrossDomainEnabled] X	600
15.3.4	getCrossDomainMessenger() [OVM_CrossDomainEnabled]	600
15.3.5	sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message) [OVM_CrossDomainEnabled]	600
15.3.6	rely(usr) X a	601
15.3.7	deny(usr) X a	601
15.3.8	constructor(_l2GovernanceRelay, _l1messenger) X	601
15.3.9	relay(target, targetData, l2gas) X a	601
15.4	contract L2DAITokenBridge	603
15.4.1	modifier onlyFromCrossDomainAccount(_sourceDomainAccount) [OVM_CrossDomainEnabled]	603
15.4.2	modifier auth()	604
15.4.3	constructor(_messenger) [OVM_CrossDomainEnabled] X	604
15.4.4	getCrossDomainMessenger() [OVM_CrossDomainEnabled]	604
15.4.5	sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message) [OVM_CrossDomainEnabled]	604
15.4.6	rely(usr) X a	605
15.4.7	deny(usr) X a	605
15.4.8	constructor(_l2CrossDomainMessenger, _l2Token, _l1Token, _l1DAITokenBridge) X	605
15.4.9	close() X a	605
15.4.10	withdraw(_l2Token, _amount, _l1Gas, _data) X	605
15.4.11	withdrawTo(_l2Token, _to, _amount, _l1Gas, _data) X	606
15.4.12	_initiateWithdrawal(_from, _to, _amount, _l1Gas, _data)	606
15.4.13	finalizeDeposit(_l1Token, _l2Token, _from, _to, _amount, _data) X	606
15.5	contract L2GovernanceRelay(2)	609
15.5.1	modifier onlyFromCrossDomainAccount(_sourceDomainAccount) [OVM_CrossDomainEnabled]	609
15.5.2	constructor(_messenger) [OVM_CrossDomainEnabled] X	610
15.5.3	getCrossDomainMessenger() [OVM_CrossDomainEnabled]	610
15.5.4	sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message) [OVM_CrossDomainEnabled]	610
15.5.5	constructor(_l2CrossDomainMessenger, _l1GovernanceRelay) X	610
15.5.6	relay(target, targetData) X	611
15.6	contract Dai(3)	613
15.6.1	modifier auth()	613
15.6.2	rely(usr) X a	613
15.6.3	deny(usr) X a	613
15.6.4	_add(x, y)	613
15.6.5	_sub(x, y)	614
15.6.6	constructor() X	614
15.6.7	_calculateDomainSeparator(chainId)	614
15.6.8	DOMAIN_SEPARATOR()	614
15.6.9	transfer(to, value) X	614
15.6.10	transferFrom(from, to, value) X	615
15.6.11	approve(spender, value) X	615
15.6.12	increaseAllowance(spender, addedValue) X	615
15.6.13	decreaseAllowance(spender, subtractedValue) X	615
15.6.14	mint(to, value) X a	616
15.6.15	burn(from, value) X	616
15.6.16	permit(owner, spender, value, deadline, v, r, s) X	616

16 StarkNet bridge	617
16.1 contract L1DAIBridge	617
16.1.1 modifier auth()	618
16.1.2 modifier whenOpen()	618
16.1.3 rely(usr) X a	618
16.1.4 deny(usr) X a	618
16.1.5 close() X a	618
16.1.6 constructor (_starkNet, _dai, _l2Dai, _escrow, _l2DaiBridge) X	618
16.1.7 setCeiling(_ceiling) X a	619
16.1.8 setMaxDeposit(_maxDeposit) X a	619
16.1.9 deposit(amount, l2Recipient) X	619
16.1.10 toSplitUint(value)	619
16.1.11 withdraw(amount, l1Recipient) X	619
16.1.12 forceWithdrawal(amount, l2Sender) X	620
16.1.13 startDepositCancellation(amount, l2Recipient, nonce) X	620
16.1.14 cancelDeposit(amount, l2Recipient, l1Recipient, nonce) X	620
16.2 contract L1GovernanceRelay(3)	621
16.2.1 modifier auth()	621
16.2.2 rely(usr) X a	621
16.2.3 deny(usr) X a	621
16.2.4 constructor (_starkNet, _l2GovernanceRelay) X	621
16.2.5 relay(spell) X a	621
16.3 contract L1DAITeleportGateway	623
16.3.1 constructor (_starkNet, _l1Token, _l2TeleportGateway, _l1Escrow, _l1TeleportRouter) X	623
16.3.2 finalizeFlush(targetDomain, daiToFlush) X	623
16.3.3 finalizeRegisterTeleport(teleport) X	623
16.3.4 toL2String(str)	624
16.3.5 toSplitUint(value)	624
16.4 contract DAIMock	625
16.4.1 constructor (name_, symbol_) [ERC20] X	626
16.4.2 name() [ERC20]	626
16.4.3 symbol() [ERC20]	626
16.4.4 decimals() [ERC20]	627
16.4.5 totalSupply() [ERC20]	627
16.4.6 balanceOf(account) [ERC20]	627
16.4.7 transfer (to, amount) [ERC20] X	627
16.4.8 allowance(owner, spender) [ERC20]	627
16.4.9 approve(spender, amount) [ERC20] X	628
16.4.10 transferFrom(from, to, amount) [ERC20] X	628
16.4.11 increaseAllowance(spender, addedValue) [ERC20] X	628
16.4.12 decreaseAllowance(spender, subtractedValue) [ERC20] X	629
16.4.13 _transfer(from, to, amount) [ERC20]	629
16.4.14 _mint(account, amount) [ERC20]	630
16.4.15 _burn(account, amount) [ERC20]	630
16.4.16 _approve(owner, spender, amount) [ERC20]	631
16.4.17 _spendAllowance(owner, spender, amount) [ERC20]	631
16.4.18 _msgSender() [Context]	631
16.4.19 _msgData() [Context]	631
16.4.20 constructor () X	632
16.5 contract TeleportRouterMock	633
16.5.1 requestMint(teleportGUID, maxFeePercentage, operatorFee) X	633
16.5.2 settle(targetDomain, batchedDaiToFlush) X	633
16.6 contract L1Escrow(3)	635
16.6.1 modifier auth()	635
16.6.2 rely(usr) X a	635
16.6.3 deny(usr) X a	635
16.6.4 constructor () X	635
16.6.5 approve(token, spender, value) X a	635
16.7 contract L1EscrowMom	637
16.7.1 modifier onlyOwner()	637
16.7.2 modifier auth()	637
16.7.3 constructor (escrow_, token_) X	637

16.7.4	isAuthorized(src, sig)	637
16.7.5	setOwner(owner_) X	637
16.7.6	setAuthority(authority_) X	638
16.7.7	refuse(spender) X a	638
17	Teleport	639
17.1	contract TeleportConstantFee	639
17.1.1	constructor(_fee, _ttl) X	639
17.1.2	getFee(guid, , , , amtToTake)	639
17.2	contract TeleportJoin	641
17.2.1	struct TeleportJoin.TeleportStatus	641
17.2.2	modifier auth()	641
17.2.3	constructor(vat_, daiJoin_, ilk_, domain_) X	641
17.2.4	_min(x, y)	642
17.2.5	rely(usr) X a	642
17.2.6	deny(usr) X a	642
17.2.7	file(what, data) X a	642
17.2.8	file(what, domain_, data) X a	642
17.2.9	file(what, domain_, data) X a	642
17.2.10	cure()	643
17.2.11	_mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee)	643
17.2.12	requestMint(teleportGUID, maxFeePercentage, operatorFee) X a	644
17.2.13	mintPending(teleportGUID, maxFeePercentage, operatorFee) X	644
17.2.14	settle(sourceDomain, batchedDaiToFlush) X	645
17.3	contract TeleportLinearFee	647
17.3.1	constructor(_fee, _ttl) X	647
17.3.2	getFee(guid, , , , amtToTake)	647
17.4	contract TeleportOracleAuth	649
17.4.1	modifier auth()	649
17.4.2	constructor(teleportJoin_) X	649
17.4.3	rely(usr) X a	649
17.4.4	deny(usr) X a	649
17.4.5	file(what, data) X a	649
17.4.6	addSigners(signers_) X a	650
17.4.7	removeSigners(signers_) X a	650
17.4.8	requestMint(teleportGUID, signatures, maxFeePercentage, operatorFee) X	650
17.4.9	isValid(signHash, signatures, threshold_)	650
17.4.10	getSignHash(teleportGUID)	651
17.4.11	splitSignature(signatures, index)	651
17.5	contract TeleportRouter	653
17.5.1	modifier auth()	653
17.5.2	constructor(dai_) X	653
17.5.3	rely(usr) X a	653
17.5.4	deny(usr) X a	653
17.5.5	file(what, domain, data) X a	653
17.5.6	numDomains()	654
17.5.7	domainAt(index)	654
17.5.8	hasDomain(domain)	654
17.5.9	requestMint(teleportGUID, maxFeePercentage, operatorFee) X	654
17.5.10	settle(targetDomain, batchedDaiToFlush) X	655
17.6	contract BasicRelay	657
17.6.1	modifier auth()	657
17.6.2	constructor(_oracleAuth, _daiJoin) X	657
17.6.3	rely(usr) X a	657
17.6.4	deny(usr) X a	657
17.6.5	addRelayers(relayers_) X a	657
17.6.6	removeRelayers(relayers_) X a	658
17.6.7	relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s) X	658
17.7	contract TrustedRelay	661
17.7.1	modifier auth()	661
17.7.2	modifier toll()	661

17.7.3	<code>constructor(_oracleAuth, _daiJoin, _ethPriceOracle)</code>	X	661
17.7.4	<code>rely(usr)</code>	X a	662
17.7.5	<code>deny(usr)</code>	X a	662
17.7.6	<code>kiss(usr)</code>	X a	662
17.7.7	<code>diss(usr)</code>	X a	662
17.7.8	<code>file(what, data)</code>	X a	662
17.7.9	<code>addRelayers(relayers_)</code>	X a	662
17.7.10	<code>removeRelayers(relayers_)</code>	X a	662
17.7.11	<code>addSigners(signers_)</code>	X	663
17.7.12	<code>removeSigners(signers_)</code>	X	663
17.7.13	<code>relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s)</code>	X	663
17.7.14	<code>requestMint(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s)</code>		664
17.7.15	<code>gasprice()</code>		664
18	Ilk registry		665
18.1	<code>contract</code> GemInfo		665
18.1.1	<code>name(token)</code>		665
18.1.2	<code>symbol(token)</code>		665
18.2	<code>contract</code> IlkRegistry		667
18.2.1	<code>struct</code> IlkRegistry.Ilk		667
18.2.2	<code>modifier</code> auth()		667
18.2.3	<code>rely(usr)</code>	X a	667
18.2.4	<code>deny(usr)</code>	X a	667
18.2.5	<code>constructor(vat_, dog_, cat_, spot_)</code>	X	668
18.2.6	<code>add(adapter)</code>	X	668
18.2.7	<code>remove(ilk)</code>	X	669
18.2.8	<code>removeAuth(ilk)</code>	X a	669
18.2.9	<code>file(what, data)</code>	X a	669
18.2.10	<code>file(ilk, what, data)</code>	X a	669
18.2.11	<code>file(ilk, what, data)</code>	X a	670
18.2.12	<code>file(ilk, what, data)</code>	X a	670
18.2.13	<code>_remove(ilk)</code>		670
18.2.14	<code>count()</code>		670
18.2.15	<code>list()</code>		670
18.2.16	<code>list(start, end)</code>		671
18.2.17	<code>get(pos)</code>		671
18.2.18	<code>info(ilk)</code>		671
18.2.19	<code>pos(ilk)</code>		671
18.2.20	<code>class(ilk)</code>		671
18.2.21	<code>gem(ilk)</code>		672
18.2.22	<code>pip(ilk)</code>		672
18.2.23	<code>join(ilk)</code>		672
18.2.24	<code>xlip(ilk)</code>		672
18.2.25	<code>dec(ilk)</code>		672
18.2.26	<code>symbol(ilk)</code>		672
18.2.27	<code>name(ilk)</code>		672
18.2.28	<code>update(ilk)</code>	X	673
18.2.29	<code>put(_ilk, _join, _gem, _dec, _class, _pip, _xlip, _name, _symbol)</code>	X a	673
18.2.30	<code>bytes32ToStr(_bytes32)</code>		674
19	DSS Vest		675
19.1	<code>contract</code> DssVestMintable		675
19.1.1	<code>struct</code> DssVest.Award		675
19.1.2	<code>modifier</code> lock() [DssVest]		676
19.1.3	<code>modifier</code> auth() [DssVest]		676
19.1.4	<code>usr(_id)</code> [DssVest]		676
19.1.5	<code>bgn(_id)</code> [DssVest]		676
19.1.6	<code>clf(_id)</code> [DssVest]		676
19.1.7	<code>fin(_id)</code> [DssVest]		676
19.1.8	<code>mgr(_id)</code> [DssVest]		676
19.1.9	<code>res(_id)</code> [DssVest]		677

19.1.10	tot(_id) [DssVest]	677
19.1.11	rxid(_id) [DssVest]	677
19.1.12	constructor() [DssVest] X	677
19.1.13	rely(_usr) [DssVest] X a	677
19.1.14	deny(_usr) [DssVest] X a	677
19.1.15	file(what, data) [DssVest] X a	677
19.1.16	min(x, y) [DssVest]	678
19.1.17	add(x, y) [DssVest]	678
19.1.18	sub(x, y) [DssVest]	678
19.1.19	mul(x, y) [DssVest]	678
19.1.20	toUint48(x) [DssVest]	678
19.1.21	toUint128(x) [DssVest]	678
19.1.22	create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a	678
19.1.23	vest(_id) [DssVest] X	679
19.1.24	vest(_id, _maxAmt) [DssVest] X	679
19.1.25	_vest(_id, _maxAmt) [DssVest]	679
19.1.26	accrued(_id) [DssVest]	680
19.1.27	accrued(_time, _bgn, _fin, _tot) [DssVest]	680
19.1.28	unpaid(_id) [DssVest]	680
19.1.29	unpaid(_time, _bgn, _clf, _fin, _tot, _rxid) [DssVest]	680
19.1.30	restrict(_id) [DssVest] X	681
19.1.31	unrestrict(_id) [DssVest] X	681
19.1.32	yank(_id) [DssVest] X	681
19.1.33	yank(_id, _end) [DssVest] X	681
19.1.34	_yank(_id, _end) [DssVest]	682
19.1.35	move(_id, _dst) [DssVest] X	682
19.1.36	valid(_id) [DssVest]	682
19.1.37	constructor(_gem) X	683
19.1.38	pay(_guy, _amt)	683
19.2	contract DssVestSuckable	685
19.2.1	struct DssVest.Award	685
19.2.2	modifier lock() [DssVest]	685
19.2.3	modifier auth() [DssVest]	686
19.2.4	usr(_id) [DssVest]	686
19.2.5	bgn(_id) [DssVest]	686
19.2.6	clf(_id) [DssVest]	686
19.2.7	fin(_id) [DssVest]	686
19.2.8	mgr(_id) [DssVest]	686
19.2.9	res(_id) [DssVest]	686
19.2.10	tot(_id) [DssVest]	686
19.2.11	rxid(_id) [DssVest]	687
19.2.12	constructor() [DssVest] X	687
19.2.13	rely(_usr) [DssVest] X a	687
19.2.14	deny(_usr) [DssVest] X a	687
19.2.15	file(what, data) [DssVest] X a	687
19.2.16	min(x, y) [DssVest]	687
19.2.17	add(x, y) [DssVest]	687
19.2.18	sub(x, y) [DssVest]	688
19.2.19	mul(x, y) [DssVest]	688
19.2.20	toUint48(x) [DssVest]	688
19.2.21	toUint128(x) [DssVest]	688
19.2.22	create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a	688
19.2.23	vest(_id) [DssVest] X	689
19.2.24	vest(_id, _maxAmt) [DssVest] X	689
19.2.25	_vest(_id, _maxAmt) [DssVest]	689
19.2.26	accrued(_id) [DssVest]	689
19.2.27	accrued(_time, _bgn, _fin, _tot) [DssVest]	690
19.2.28	unpaid(_id) [DssVest]	690
19.2.29	unpaid(_time, _bgn, _clf, _fin, _tot, _rxid) [DssVest]	690
19.2.30	restrict(_id) [DssVest] X	690
19.2.31	unrestrict(_id) [DssVest] X	691
19.2.32	yank(_id) [DssVest] X	691
19.2.33	yank(_id, _end) [DssVest] X	691

19.2.34	<code>_yank(_id, _end)</code> [DssVest]	691
19.2.35	<code>move(_id, _dst)</code> [DssVest] X	692
19.2.36	<code>valid(_id)</code> [DssVest]	692
19.2.37	<code>constructor(_chainlog)</code> X	692
19.2.38	<code>pay(_guy, _amt)</code>	693
19.3	<code>contract</code> DssVestTransferrable	695
19.3.1	<code>struct</code> DssVest.Award	695
19.3.2	<code>modifier</code> lock() [DssVest]	695
19.3.3	<code>modifier</code> auth() [DssVest]	696
19.3.4	<code>usr(_id)</code> [DssVest]	696
19.3.5	<code>bgn(_id)</code> [DssVest]	696
19.3.6	<code>clf(_id)</code> [DssVest]	696
19.3.7	<code>fin(_id)</code> [DssVest]	696
19.3.8	<code>mgr(_id)</code> [DssVest]	696
19.3.9	<code>res(_id)</code> [DssVest]	696
19.3.10	<code>tot(_id)</code> [DssVest]	697
19.3.11	<code>rxid(_id)</code> [DssVest]	697
19.3.12	<code>constructor()</code> [DssVest] X	697
19.3.13	<code>rely(_usr)</code> [DssVest] X a	697
19.3.14	<code>deny(_usr)</code> [DssVest] X a	697
19.3.15	<code>file(what, data)</code> [DssVest] X a	697
19.3.16	<code>min(x, y)</code> [DssVest]	697
19.3.17	<code>add(x, y)</code> [DssVest]	698
19.3.18	<code>sub(x, y)</code> [DssVest]	698
19.3.19	<code>mul(x, y)</code> [DssVest]	698
19.3.20	<code>toUint48(x)</code> [DssVest]	698
19.3.21	<code>toUint128(x)</code> [DssVest]	698
19.3.22	<code>create(_usr, _tot, _bgn, _tau, _eta, _mgr)</code> [DssVest] X a	698
19.3.23	<code>vest(_id)</code> [DssVest] X	699
19.3.24	<code>vest(_id, _maxAmt)</code> [DssVest] X	699
19.3.25	<code>_vest(_id, _maxAmt)</code> [DssVest]	699
19.3.26	<code>accrued(_id)</code> [DssVest]	700
19.3.27	<code>accrued(_time, _bgn, _fin, _tot)</code> [DssVest]	700
19.3.28	<code>unpaid(_id)</code> [DssVest]	700
19.3.29	<code>unpaid(_time, _bgn, _clf, _fin, _tot, _rxid)</code> [DssVest]	700
19.3.30	<code>restrict(_id)</code> [DssVest] X	701
19.3.31	<code>unrestrict(_id)</code> [DssVest] X	701
19.3.32	<code>yank(_id)</code> [DssVest] X	701
19.3.33	<code>yank(_id, _end)</code> [DssVest] X	701
19.3.34	<code>_yank(_id, _end)</code> [DssVest]	701
19.3.35	<code>move(_id, _dst)</code> [DssVest] X	702
19.3.36	<code>valid(_id)</code> [DssVest]	702
19.3.37	<code>constructor(_czar, _gem)</code> X	703
19.3.38	<code>pay(_guy, _amt)</code>	703
20	Unassigned	705
20.1	<code>contract</code> DssAutoLine	705
20.1.1	<code>struct</code> DssAutoLine.Ilkc	705
20.1.2	<code>modifier</code> auth()	705
20.1.3	<code>constructor(vat_)</code> X	705
20.1.4	<code>add(x, y)</code>	706
20.1.5	<code>sub(x, y)</code>	706
20.1.6	<code>mul(x, y)</code>	706
20.1.7	<code>min(x, y)</code>	706
20.1.8	<code>setIlkc(ilkc, line, gap, ttl)</code> X a	706
20.1.9	<code>remIlkc(ilkc)</code> X a	706
20.1.10	<code>rely(usr)</code> X a	707
20.1.11	<code>deny(usr)</code> X a	707
20.1.12	<code>exec(_ilkc)</code> X	707
20.2	<code>contract</code> ChainLog	709
20.2.1	<code>struct</code> ChainLog.Location	709
20.2.2	<code>modifier</code> auth()	709
20.2.3	<code>rely(usr)</code> X a	709

20.2.4	deny(usr) X a	709
20.2.5	constructor() X	709
20.2.6	setVersion(_version) X a	710
20.2.7	setSha256sum(_sha256sum) X a	710
20.2.8	setIPFS(_ipfs) X a	710
20.2.9	setAddress(_key, _addr) X a	710
20.2.10	removeAddress(_key) X a	710
20.2.11	count()	711
20.2.12	get(_index)	711
20.2.13	list()	711
20.2.14	getAddress(_key)	711
20.2.15	_addAddress(_key, _addr)	711
20.2.16	_removeAddress(_key)	711

21 Dependencies 713

21.1	library ScriptTools	713
21.1.1	getRootChainId()	713
21.1.2	readInput(name)	713
21.1.3	readOutput(name, timestamp)	713
21.1.4	readOutput(name)	714
21.1.5	loadConfig(name)	714
21.1.6	loadConfig()	714
21.1.7	loadDependencies(name)	714
21.1.8	loadDependencies()	715
21.1.9	exportContract(name, label, addr)	715
21.1.10	exportContract(label, addr)	715
21.1.11	stringToBytes32(source)	715
21.1.12	ilkToChainlogFormat(ilk)	716
21.1.13	eq(a, b)	716
21.1.14	switchOwner(base, deployer, newOwner)	716
21.1.15	readUint(json, key, envKey)	716
21.1.16	readUintArray(json, key, envKey)	717
21.1.17	readInt(json, key, envKey)	717
21.1.18	readIntArray(json, key, envKey)	717
21.1.19	readBytes32(json, key, envKey)	717
21.1.20	readBytes32Array(json, key, envKey)	717
21.1.21	readString(json, key, envKey)	717
21.1.22	readStringArray(json, key, envKey)	717
21.1.23	readAddress(json, key, envKey)	717
21.1.24	readAddressArray(json, key, envKey)	718
21.1.25	readBool(json, key, envKey)	718
21.1.26	readBoolArray(json, key, envKey)	718
21.1.27	readBytes(json, key, envKey)	718
21.1.28	readBytesArray(json, key, envKey)	718
21.2	library MCD	719
21.2.1	getAddressOrNull(dss, key)	719
21.2.2	loadFromChainlog(chainlog)	719
21.2.3	loadFromChainlog(chainlog)	719
21.2.4	bytesToBytes32(b)	719
21.2.5	getIlk(dss, gem, variant)	720
21.2.6	initIlk(dss, ilk)	720
21.2.7	initIlk(dss, ilk, join)	720
21.2.8	initIlk(dss, ilk, join, pip)	720
21.2.9	initIlk(dss, ilk, join, pip, clip, clipCalc)	720
21.2.10	giveAdminAccess(dss, who)	721
21.2.11	giveAdminAccess(dss)	721
21.2.12	newUser(dss)	721
21.3	library D3MDeploy	723
21.3.1	deployCore(deployer, owner, daiJoin)	723
21.3.2	deployAave(deployer, owner, ilk, vat, hub, dai, lendingPool)	723
21.3.3	deployCompound(deployer, owner, ilk, vat, hub, cdai)	723
21.4	library D3MInit	725
21.4.1	initCore(dss, d3mCore)	725

21.4.2	_init(dss, d3m, cfg, gem)	725
21.4.3	initAave(dss, d3m, cfg, aaveCfg)	726
21.4.4	initCompound(dss, d3m, cfg, compoundCfg)	727
21.5	library DssExecLib	729
21.5.1	wdiv(x, y)	730
21.5.2	rdiv(x, y)	730
21.5.3	dai()	730
21.5.4	mkr()	730
21.5.5	vat()	730
21.5.6	cat()	730
21.5.7	dog()	730
21.5.8	jug()	731
21.5.9	pot()	731
21.5.10	vow()	731
21.5.11	end()	731
21.5.12	esm()	731
21.5.13	reg()	731
21.5.14	spotter()	731
21.5.15	flap()	731
21.5.16	flop()	731
21.5.17	osmMom()	731
21.5.18	govGuard()	732
21.5.19	flipperMom()	732
21.5.20	clipperMom()	732
21.5.21	pauseProxy()	732
21.5.22	autoLine()	732
21.5.23	daiJoin()	732
21.5.24	lerpFab()	732
21.5.25	clip(_ilk)	732
21.5.26	flip(_ilk)	732
21.5.27	calc(_ilk)	732
21.5.28	getChangelogAddress(_key)	733
21.5.29	setChangelogAddress(_key, _val) X	733
21.5.30	setChangelogVersion(_version) X	733
21.5.31	setChangelogIPFS(_ipfsHash) X	733
21.5.32	setChangelogSHA256(_SHA256Sum) X	733
21.5.33	authorize(_base, _ward) X	733
21.5.34	deauthorize(_base, _ward) X	734
21.5.35	setAuthority(_base, _authority) X	734
21.5.36	delegateVat(_usr) X	734
21.5.37	undelegateVat(_usr) X	734
21.5.38	canCast(_ts, _officeHours)	734
21.5.39	nextCastTime(_eta, _ts, _officeHours)	735
21.5.40	accumulateDSR() X	735
21.5.41	accumulateCollateralStabilityFees(_ilk) X	735
21.5.42	updateCollateralPrice(_ilk) X	736
21.5.43	setContract(_base, _what, _addr) X	736
21.5.44	setContract(_base, _ilk, _what, _addr) X	736
21.5.45	setValue(_base, _what, _amt) X	736
21.5.46	setValue(_base, _ilk, _what, _amt) X	736
21.5.47	setGlobalDebtCeiling(_amount) X	737
21.5.48	increaseGlobalDebtCeiling(_amount) X	737
21.5.49	decreaseGlobalDebtCeiling(_amount) X	737
21.5.50	setDSR(_rate, _doDrip) X	737
21.5.51	setSurplusAuctionAmount(_amount) X	738
21.5.52	setSurplusBuffer(_amount) X	738
21.5.53	setMinSurplusAuctionBidIncrease(_pct_bps) X	738
21.5.54	setSurplusAuctionBidDuration(_duration) X	738
21.5.55	setSurplusAuctionDuration(_duration) X	738
21.5.56	setDebtAuctionDelay(_duration) X	739
21.5.57	setDebtAuctionDAIAmount(_amount) X	739
21.5.58	setDebtAuctionMKRAmount(_amount) X	739
21.5.59	setMinDebtAuctionBidIncrease(_pct_bps) X	739

21.5.60	setDebtAuctionBidDuration(_duration) X	739
21.5.61	setDebtAuctionDuration(_duration) X	740
21.5.62	setDebtAuctionMKRIncreaseRate(_pct_bps) X	740
21.5.63	setMaxTotalDAILiquidationAmount(_amount) X	740
21.5.64	setMaxTotalDAILiquidationAmountLEGACY(_amount) X	740
21.5.65	setEmergencyShutdownProcessingTime(_duration) X	740
21.5.66	setGlobalStabilityFee(_rate) X	741
21.5.67	setDAIReferenceValue(_value) X	741
21.5.68	setIlkDebtCeiling(_ilk, _amount) X	741
21.5.69	increaseIlkDebtCeiling(_ilk, _amount, _global) X	741
21.5.70	decreaseIlkDebtCeiling(_ilk, _amount, _global) X	742
21.5.71	setRWAILkDebtCeiling(_ilk, _ceiling, _price) X	742
21.5.72	setIlkAutoLineParameters(_ilk, _amount, _gap, _ttl) X	742
21.5.73	setIlkAutoLineDebtCeiling(_ilk, _amount) X	743
21.5.74	removeIlkFromAutoLine(_ilk) X	743
21.5.75	setIlkMinVaultAmount(_ilk, _amount) X	743
21.5.76	setIlkLiquidationPenalty(_ilk, _pct_bps) X	743
21.5.77	setIlkMaxLiquidationAmount(_ilk, _amount) X	744
21.5.78	setIlkLiquidationRatio(_ilk, _pct_bps) X	744
21.5.79	setStartingPriceMultiplicativeFactor(_ilk, _pct_bps) X	744
21.5.80	setAuctionTimeBeforeReset(_ilk, _duration) X	745
21.5.81	setAuctionPermittedDrop(_ilk, _pct_bps) X	745
21.5.82	setKeeperIncentivePercent(_ilk, _pct_bps) X	745
21.5.83	setKeeperIncentiveFlatRate(_ilk, _amount) X	745
21.5.84	setLiquidationBreakerPriceTolerance(_clip, _pct_bps) X	745
21.5.85	setIlkStabilityFee(_ilk, _rate, _doDrip) X	746
21.5.86	setLinearDecrease(_calc, _duration) X	746
21.5.87	setStairstepExponentialDecrease(_calc, _duration, _pct_bps) X	746
21.5.88	setExponentialDecrease(_calc, _pct_bps) X	747
21.5.89	whitelistOracleMedians(_oracle) X	747
21.5.90	addReaderToWhitelist(_oracle, _reader) X	747
21.5.91	removeReaderFromWhitelist(_oracle, _reader) X	747
21.5.92	addReaderToWhitelistCall(_oracle, _reader) X	748
21.5.93	removeReaderFromWhitelistCall(_oracle, _reader) X	748
21.5.94	setMedianWritersQuorum(_median, _minQuorum) X	748
21.5.95	allowOSMFreeze(_osm, _ilk) X	748
21.5.96	setD3MTargetInterestRate(_d3m, _pct_bps) X	749
21.5.97	addCollateralBase(_ilk, _gem, _join, _clip, _calc, _pip) X	749
21.5.98	addNewCollateral(co) X	750
21.5.99	sendPaymentFromSurplusBuffer(_target, _amount) X	751
21.5.100	linearInterpolation(_name, _target, _what, _startTime, _start, _end, _duration) X	751
21.5.101	linearInterpolation(_name, _target, _ilk, _what, _startTime, _start, _end, _duration) X	752
21.6	library SafeMath	753
21.6.1	mul(a, b)	753
21.6.2	div(a, b)	753
21.6.3	sub(a, b)	753
21.6.4	add(a, b)	753
21.7	library EnumerableSet	755
21.7.1	struct EnumerableSet.Set	755
21.7.2	struct EnumerableSet.Bytes32Set	755
21.7.3	struct EnumerableSet.AddressSet	755
21.7.4	struct EnumerableSet.UintSet	756
21.7.5	_add(set, value)	756
21.7.6	_remove(set, value)	756
21.7.7	_contains(set, value)	757
21.7.8	_length(set)	757
21.7.9	_at(set, index)	757
21.7.10	_values(set)	757
21.7.11	add(set, value)	758
21.7.12	remove(set, value)	758
21.7.13	contains(set, value)	758

21.7.14	<code>length(set)</code>	758
21.7.15	<code>at(set, index)</code>	758
21.7.16	<code>values(set)</code>	759
21.7.17	<code>add(set, value)</code>	759
21.7.18	<code>remove(set, value)</code>	759
21.7.19	<code>contains(set, value)</code>	759
21.7.20	<code>length(set)</code>	759
21.7.21	<code>at(set, index)</code>	760
21.7.22	<code>values(set)</code>	760
21.7.23	<code>add(set, value)</code>	760
21.7.24	<code>remove(set, value)</code>	760
21.7.25	<code>contains(set, value)</code>	761
21.7.26	<code>length(set)</code>	761
21.7.27	<code>at(set, index)</code>	761
21.7.28	<code>values(set)</code>	761
21.8	<code>library</code> Address	763
21.8.1	<code>isContract(account)</code>	763
21.8.2	<code>sendValue(recipient, amount)</code>	763
21.8.3	<code>functionCall(target, data)</code>	764
21.8.4	<code>functionCall(target, data, errorMessage)</code>	764
21.8.5	<code>functionCallWithValue(target, data, value)</code>	764
21.8.6	<code>functionCallWithValue(target, data, value, errorMessage)</code>	765
21.8.7	<code>functionStaticCall(target, data)</code>	765
21.8.8	<code>functionStaticCall(target, data, errorMessage)</code>	765
21.8.9	<code>functionDelegateCall(target, data)</code>	765
21.8.10	<code>functionDelegateCall(target, data, errorMessage)</code>	766
21.8.11	<code>_verifyCallResult(success, returndata, errorMessage)</code>	766
21.9	<code>library</code> GodMode	767
21.9.1	<code>vm()</code>	767
21.9.2	<code>setWard(base, target, val)</code>	767
21.9.3	<code>setWard(base, target, val)</code>	767
21.9.4	<code>setWard(base, target, val)</code>	768
21.9.5	<code>setBalance(token, who, amount)</code>	768
21.9.6	<code>setBalance(token, who, amount)</code>	769
21.9.7	<code>setBalance(token, who, amount)</code>	769

References index

A

AAVE

definition

`contract` AAVE, 451

aaveCfg

write

`D3MInitScript.run()` **X**, 420

definition

`D3MInitScript.aaveCfg`: `D3MAaveConfig`, 403

read

`D3MInitScript.run()` **X**, 420

AccessList

definition

`struct` `StdCheatsSafe.AccessList`, 333, 361, 387, 407

accrued

call

`DssVestMintable.accrued(_id)` `[DssVest]`, 680

`DssVestMintable.unpaid(_time, _bgn, _clf, _fin, _tot, _rxn)` `[DssVest]`, 680

`DssVestSuckable.accrued(_id)` `[DssVest]`, 689

`DssVestSuckable.unpaid(_time, _bgn, _clf, _fin, _tot, _rxn)` `[DssVest]`, 690

`DssVestTransferrable.accrued(_id)` `[DssVest]`, 700

`DssVestTransferrable.unpaid(_time, _bgn, _clf, _fin, _tot, _rxn)` `[DssVest]`, 700

definition

`DssVestMintable.accrued(_id)` `[DssVest]`, 680

`DssVestMintable.accrued(_time, _bgn, _fin, _tot)` `[DssVest]`, 680

`DssVestSuckable.accrued(_id)` `[DssVest]`, 689

`DssVestSuckable.accrued(_time, _bgn, _fin, _tot)` `[DssVest]`, 690

`DssVestTransferrable.accrued(_id)` `[DssVest]`, 700

`DssVestTransferrable.accrued(_time, _bgn, _fin, _tot)` `[DssVest]`, 700

accrueInterest

call

`D3MCompoundPool.preDebtChange()` **X**, 379

accumulateCollateralStabilityFees

definition

`DssExecLib.accumulateCollateralStabilityFees(_ilk)` **X**, 735

accumulateDSR

definition

`DssExecLib.accumulatedDSR()` **X**, 735

action

write

`DssExec.constructor(_expiration, _spellAction)` **X**, 309

definition

`DssExec.action`: `address`, 309

read

`DssExec.cast()` **X**, 310

`DssExec.description()`, 309

`DssExec.nextCastTime()`, 309

`DssExec.officeHours()`, 309

`DssExec.schedule()` **X**, 309

actions

call

`DssAction.execute()` **X**, 307

active

call

`D3MAavePlan.disable()` **X**, 314

`D3MCompoundPlan.disable()` **X**, 317

`D3MHub._exec(ilk, _pool, Art, lineWad)`, 322

write

`Clipper.kick(tab, lot, usr, kpr)` **X** **a**, 244

`Clipper._remove(id)`, 247

definition

`Clipper.active`: `uint256[]`, 241

`D3MAavePlan.active()`, 314

```

    D3MCompoundPlan.active(), 317
read
    Clipper.count(), 248
    Clipper.kick(tab, lot, usr, kpr) X a, 244
    Clipper.list(), 248
    Clipper.remove(id), 247
activeFork
    call
        D3MCoreDeployScript.isFork() [StdCheatsSafe], 343
        D3MCoreInitScript.isFork() [StdCheatsSafe], 371
        D3MDeployScript.isFork() [StdCheatsSafe], 397
        D3MInitScript.isFork() [StdCheatsSafe], 417
activeOutbox
    call
        L1DaiGateway.modifier onlyL2Counterpart(l2Counterpart) [L1CrossDomainEnabled], 569
        L1GovernanceRelay.modifier onlyL2Counterpart(l2Counterpart)
            [L1CrossDomainEnabled], 577
ada
    write
        GemBag.constructor(lad_, gem_) X, 431
    definition
        GemBag.ada: address, 431
    read
        GemBag.exit(usr, wad) X, 431
adai
    call
        D3MInit.initAave(dss, d3m, cfg, aaveCfg), 726
        D3MInitScript.run() X, 420
    write
        D3MAavePlan.constructor(dai_, pool_) X, 311
        D3MAavePool.constructor(ilk_, hub_, dai_, pool_) X, 349
    definition
        D3MAavePlan.adai: address, 311
        D3MAavePool.adai: ATokenLike, 349
    read
        D3MAavePlan.active(), 314
        D3MAavePlan.getTargetAssets(currentAssets), 313
        D3MAavePool.assetBalance(), 351
        D3MAavePool.collect() X, 352
        D3MAavePool.deposit(wad) X, 350
        D3MAavePool.exit(dst, wad) X, 351
        D3MAavePool.maxWithdraw(), 351
        D3MAavePool.quit(dst) X a, 351
        D3MAavePool.redeemable(), 352
adaiRevision
    call
        D3MInit.initAave(dss, d3m, cfg, aaveCfg), 726
        D3MInitScript.run() X, 420
    write
        D3MAavePlan.constructor(dai_, pool_) X, 311
    definition
        D3MAavePlan.adaiRevision: uint256, 311
    read
        D3MAavePlan.active(), 314
.add
    call
        AAVE.transferFrom(src, dst, wad) X, 452
        BAL.transferFrom(src, dst, wad) X, 454
        BAT.transferFrom(src, dst, wad) X, 456
        CalleeMakerOtcDai.clipperCall(sender, daiAmt, gemAmt, data) X, 550
        CalleeMakerOtcDai.divup(x, y) [CalleeMakerOtc], 549
        Clipper.kick(tab, lot, usr, kpr) X a, 244
        ClipperMom.setBreaker(clip, level, delay) X a, 258

```

Clipper.redo(id, kpr) [X](#), 245
 COMP.transferFrom(src, dst, wad) [X](#), 458
 Cure.cage() [X](#) [a](#), 293
 Cure.load(src) [X](#), 293
 CurveLpTokenUniv3Callee.clipperCall(sender, owe, slice, data) [X](#), 552
 CurveLpTokenUniv3Callee.divup(x, y), 551
 Dai.increaseAllowance(spender, addedValue) [X](#), 589, 615
 Dai.mint(to, value) [X](#) [a](#), 590, 616
 Dai.mint(usr, wad) [X](#) [a](#), 224
 Dai.transferFrom(src, dst, wad) [X](#), 224
 DGD.transferFrom(src, dst, wad) [X](#), 460
 Dog.bark(ilk, urn, kpr) [X](#), 253
 DssAutoLine.exec(_ilk) [X](#), 707
 DssCdpManager.give(cdp, dst) [X](#), 523
 DssCdpManager.open(ilk, usr) [X](#), 522
 DssExecLib.addCollateralBase(_ilk, _gem, _join, _clip, _calc, _pip) [X](#), 749
 DssPsm.buyGem(usr, gemAmt) [X](#), 511
 DssVestMintable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] [X](#) [a](#), 678
 DssVestMintable._vest(_id, _maxAmt) [DssVest], 679
 DssVestMintable._yank(_id, _end) [DssVest], 682
 DssVestSuckable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] [X](#) [a](#), 688
 DssVestSuckable._vest(_id, _maxAmt) [DssVest], 689
 DssVestSuckable._yank(_id, _end) [DssVest], 691
 DssVestTransferrable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] [X](#) [a](#), 698
 DssVestTransferrable._vest(_id, _maxAmt) [DssVest], 699
 DssVestTransferrable._yank(_id, _end) [DssVest], 701
 End.cash(ilk, wad) [X](#), 301
 End.pack(wad) [X](#), 301
 End.skim(ilk, urn) [X](#), 300
 End.skip(ilk, id) [X](#), 299
 End.snip(ilk, id) [X](#), 299
 End.thaw() [X](#), 300
 EnumerableSet.add(set, value), 758–760
 ESM.join(wad) [X](#), 305
 Flapper.kick(lot, bid) [X](#) [a](#), 263
 Flapper.tend(id, lot, bid) [X](#), 263
 Flapper.tick(id) [X](#), 263
 Flopper.dent(id, lot, bid) [X](#), 267
 Flopper.kick(gal, lot, bid) [X](#) [a](#), 266
 Flopper.tick(id) [X](#), 267
 GemJoin9._join(usr), 444
 GNT.transfer(dst, wad) [X](#), 461
 GUSD.transferFrom(src, dst, wad) [X](#), 464
 Jug.drip(ilk) [X](#), 275
 KNC.transferFrom(src, dst, wad) [X](#), 466
 LINK.transferFrom(src, dst, wad) [X](#), 468
 LRC.transferFrom(src, dst, wad) [X](#), 470
 MANA.transferFrom(src, dst, wad) [X](#), 472
 MATIC.transferFrom(src, dst, wad) [X](#), 474
 OMG.transferFrom(src, dst, wad) [X](#), 476
 OSM.pass(), 287
 PAXG.transferFrom(src, dst, wad) [X](#), 478
 PAXUSD.transferFrom(src, dst, wad) [X](#), 482
 Pot.join(wad) [X](#), 279
 PSMCallee.clipperCall(sender, owe, slice, data) [X](#), 555
 PSMCallee.divup(x, y), 555
 RENBTC.transferFrom(src, dst, wad) [X](#), 484
 REP.transferFrom(src, dst, wad) [X](#), 486
 TeleportRouter.file(what, domain, data) [X](#) [a](#), 653
 TUSD.transferFrom(src, dst, wad) [X](#), 488
 UniswapV2CalleeDai.clipperCall(sender, daiAmt, gemAmt, data) [X](#), 558
 UniswapV2CalleeDai.divup(x, y) [UniswapV2Callee], 557
 UniswapV2LpTokenCalleeDai.clipperCall(sender, daiAmt, gemAmt, data) [X](#), 560

UniswapV2LpTokenCalleeDai.divup(x, y) [UniswapV2Callee(2)], 559
 UniswapV3Callee.clipperCall(sender, owe, slice, data) X, 567
 UniswapV3Callee.divup(x, y), 567
 UNI.transferFrom(src, dst, wad) X, 490
 USDC.transferFrom(src, dst, wad) X, 492
 USDT.transfer(_to, _value) X, 493
 USDT.transferFrom(_from, _to, _value) X, 494
 Vat.flux(ilk, src, dst, wad) X, 231
 Vat.fold(i, u, rate) X a, 234
 Vat.fork(ilk, src, dst, dink, dart) X, 233
 Vat.frob(i, u, v, w, dink, dart) X, 232
 Vat.grab(i, u, v, w, dink, dart) X a, 233
 Vat.move(src, dst, rad) X, 232
 Vat.slip(ilk, usr, wad) X a, 231
 Vat.suck(u, v, rad) X a, 234
 Vow.fess(tab) X a, 270
 Vow.flap() X, 271
 Vow.flog(era) X, 270
 Vow.flop() X, 271
 WBTC.transferFrom(src, dst, wad) X, 496
 WstETHCurveUniv3Callee.clipperCall(sender, owe, slice, data) X, 564
 WstETHCurveUniv3Callee.divup(x, y), 563
 WSTETH.transferFrom(src, dst, wad) X, 498
 YFI.transferFrom(src, dst, wad) X, 500
 ZRX.transferFrom(src, dst, wad) X, 502

definition

AAVE.add(x, y), 451
 BAL.add(x, y), 453
 BAT.add(x, y), 455
 CalleeMakerOtcDai.add(x, y) [CalleeMakerOtc], 549
 Clipper.add(x, y), 243
 ClipperMom.add(x, y), 257
 COMP.add(x, y), 457
 Cure.add(x, y), 291
 CurveLpTokenUniv3Callee.add(x, y), 551
 Dai.add(x, y), 224, 587, 613
 DGD.add(x, y), 459
 Dog.add(x, y), 252
 DssAutoLine.add(x, y), 706
 DssCdpManager.add(x, y), 522
 DssPsm.add(x, y), 510
 DssVestMintable.add(x, y) [DssVest], 678
 DssVestSuckable.add(x, y) [DssVest], 687
 DssVestTransferrable.add(x, y) [DssVest], 698
 End.add(x, y), 297
 EnumerableSet.add(set, value), 756, 758–760
 ESM.add(x, y), 303
 Flapper.add(x, y), 262
 Flopper.add(x, y), 266
 GemJoin9.add(x, y), 444
 GNT.add(x, y), 461
 GUSD.add(x, y), 463
 IlkRegistry.add(adapter) X, 668
 Jug.add(x, y), 274
 KNC.add(x, y), 465
 LinearDecrease.add(x, y), 236
 LINK.add(x, y), 467
 LRC.add(x, y), 469
 MANA.add(x, y), 471
 MATIC.add(x, y), 473
 OMG.add(x, y), 475
 OSM.add(x, y), 286
 PAXG.add(x, y), 477

PAXUSD.add(x, y), 481
 Pot._add(x, y), 278
 PSMCallee._add(x, y), 555
 RENBTC.add(x, y), 483
 REP.add(x, y), 485
 SafeMath.add(a, b), 753
 TUSD.add(x, y), 487
 UNI.add(x, y), 489
 UniswapV2CalleeDai.add(x, y) [UniswapV2Callee], 557
 UniswapV2LpTokenCalleeDai.add(x, y) [UniswapV2Callee(2)], 559
 UniswapV3Callee._add(x, y), 567
 USDC.add(x, y), 491
 Vat._add(x, y), 230
 Vow.add(x, y), 269
 WBTC.add(x, y), 495
 WSTETH.add(x, y), 497
 WstETHCurveUniv3Callee._add(x, y), 563
 YFI.add(x, y), 499
 ZRX.add(x, y), 501

add256
 call
 Flapper.kick(lot, bid) X a, 263
 definition
 Flapper.add256(x, y), 262

_addAddress
 call
 ChainLog.setAddress(_key, _addr) X a, 710
 definition
 ChainLog._addAddress(_key, _addr), 711

addCollateralBase
 call
 DssExecLib.addNewCollateral(co) X, 750
 definition
 DssExecLib.addCollateralBase(_ilk, _gem, _join, _clip, _calc, _pip) X, 749

addNewCollateral
 definition
 DssExecLib.addNewCollateral(co) X, 750

addr
 call
 D3MCoreDeployScript.makeAddrAndKey(name) [StdCheatsSafe], 342
 D3MCoreInitScript.makeAddrAndKey(name) [StdCheatsSafe], 370
 D3MDeployScript.makeAddrAndKey(name) [StdCheatsSafe], 396
 D3MInitScript.makeAddrAndKey(name) [StdCheatsSafe], 417
 write
 ChainLog.setAddress(_key, _addr) X a, 710
 read
 ChainLog.get(_index), 711
 ChainLog.getAddress(_key), 711

addReaderToWhitelist
 call
 DssExecLib.addNewCollateral(co) X, 750
 definition
 DssExecLib.addReaderToWhitelist(_oracle, _reader) X, 747

addReaderToWhitelistCall
 call
 DssExecLib.whitelistOracleMedians(_oracle) X, 747
 definition
 DssExecLib.addReaderToWhitelistCall(_oracle, _reader) X, 748

addRelayers
 definition
 BasicRelay.addRelayers(relayers_) X a, 657
 TrustedRelay.addRelayers(relayers_) X a, 662

Address

definition
 library Address, 763
 addressFromLast20Bytes
 call
 D3MCoreDeployScript.computeCreate2Address(salt, initcodeHash, deployer) [StdUtils], 337
 D3MCoreDeployScript.computeCreateAddress(deployer, nonce) [StdUtils], 336
 D3MCoreInitScript.computeCreate2Address(salt, initcodeHash, deployer) [StdUtils], 365
 D3MCoreInitScript.computeCreateAddress(deployer, nonce) [StdUtils], 364
 D3MDeployScript.computeCreate2Address(salt, initcodeHash, deployer) [StdUtils], 391
 D3MDeployScript.computeCreateAddress(deployer, nonce) [StdUtils], 390
 D3MInitScript.computeCreate2Address(salt, initcodeHash, deployer) [StdUtils], 411
 D3MInitScript.computeCreateAddress(deployer, nonce) [StdUtils], 410
 definition
 D3MCoreDeployScript.addressFromLast20Bytes(bytesValue) [StdUtils], 337
 D3MCoreInitScript.addressFromLast20Bytes(bytesValue) [StdUtils], 365
 D3MDeployScript.addressFromLast20Bytes(bytesValue) [StdUtils], 391
 D3MInitScript.addressFromLast20Bytes(bytesValue) [StdUtils], 412
 AddressSet
 definition
 struct EnumerableSet.AddressSet, 755
 addSigners
 definition
 TeleportOracleAuth.addSigners(signers_) X a, 650
 TrustedRelay.addSigners(signers_) X, 663
 admin
 write
 D3MCoreDeployScript.run() X, 346
 D3MDeployScript.run() X, 400
 definition
 D3MCoreDeployScript.admin: address, 329
 D3MDeployScript.admin: address, 383
 read
 D3MCoreDeployScript.run() X, 346
 D3MDeployScript.run() X, 400
 .afterTokenTransfer
 call
 DAIMock.burn(account, amount) [ERC20], 630
 DAIMock.mint(account, amount) [ERC20], 630
 DAIMock.transfer(from, to, amount) [ERC20], 629
 age
 write
 Median.poke(val_, age_, v, r, s) X, 283
 definition
 Median.age: uint32, 281
 read
 Median.poke(val_, age_, v, r, s) X, 283
 allDomains
 definition
 TeleportRouter.allDomains: EnumerableSet.Bytes32Set, 653
 read
 TeleportRouter.domainAt(index), 654
 TeleportRouter.file(what, domain, data) X a, 653
 TeleportRouter.hasDomain(domain), 654
 TeleportRouter.numDomains(), 654
 allowance
 call
 DAIMock.decreaseAllowance(spender, subtractedValue) [ERC20] X, 629
 DAIMock.increaseAllowance(spender, addedValue) [ERC20] X, 628
 DAIMock.spendAllowance(owner, spender, amount) [ERC20], 631
 GemJoin2.join(usr, wad) X, 428

write

Dai.approve(spender, value) X, 589, 615
 Dai.approve(usr, wad) X, 225
 Dai.burn(from, value) X, 590, 616
 Dai.burn(usr, wad) X, 225
 Dai.decreaseAllowance(spender, subtractedValue) X, 589, 615
 Dai.increaseAllowance(spender, addedValue) X, 589, 615
 Dai.permit(holder, spender, nonce, expiry, allowed, v, r, s) X, 225
 Dai.permit(owner, spender, value, deadline, v, r, s) X, 590, 616
 Dai.transferFrom(from, to, value) X, 589, 615
 Dai.transferFrom(src, dst, wad) X, 224
 WETH9.approve(guy, wad) X, 517
 WETH9.transferFrom(src, dst, wad) X, 518

definition

AAVE.allowance(src, guy), 451
 BAL.allowance(src, guy), 453
 BAT.allowance(src, guy), 455
 COMP.allowance(src, guy), 457
 Dai.allowance: mapping(address => mapping(address => uint256)), 223, 587, 613
 DAIMock.allowance(owner, spender) [ERC20], 627
 DGD.allowance(src, guy), 459
 GUSD.allowance(src, guy), 463
 KNC.allowance(src, guy), 465
 LINK.allowance(src, guy), 467
 LRC.allowance(src, guy), 469
 MANA.allowance(src, guy), 471
 MATIC.allowance(src, guy), 473
 OMG.allowance(src, guy), 475
 PAXG.allowance(src, guy), 478
 PAXUSD.allowance(src, guy), 481
 RENBTC.allowance(src, guy), 483
 REP.allowance(src, guy), 485
 TUSD.allowance(src, guy), 487
 UNI.allowance(src, guy), 489
 USDC.allowance(src, guy), 491
 USDT.allowance(_owner, _spender), 494
 WBTC.allowance(src, guy), 495
 WETH9.allowance: mapping(address => mapping(address => uint256)), 517
 WSTETH.allowance(src, guy), 497
 YFI.allowance(src, guy), 499
 ZRX.allowance(src, guy), 501

read

Dai.burn(from, value) X, 590, 616
 Dai.burn(usr, wad) X, 225
 Dai.decreaseAllowance(spender, subtractedValue) X, 589, 615
 Dai.increaseAllowance(spender, addedValue) X, 589, 615
 Dai.transferFrom(from, to, value) X, 589, 615
 Dai.transferFrom(src, dst, wad) X, 224
 WETH9.transferFrom(src, dst, wad) X, 518

_allowances**write**

DAIMock._approve(owner, spender, amount) [ERC20], 631

definition

DAIMock._allowances [ERC20] : mapping(address => mapping(address => uint256)), 625

read

DAIMock.allowance(owner, spender) [ERC20], 627

allowed**write**

USDT.approve(_spender, _value) X, 494
 USDT.transferFrom(_from, _to, _value) X, 494

definition

USDT.allowed: mapping(address => mapping(address => uint256)), 493

read
 USDT.allowance(_owner, _spender), 494
 USDT.approve(_spender, _value) [X](#), 494
 USDT.transferFrom(_from, _to, _value) [X](#), 494

allowOSMFreeze
 call
 DssExecLib.addNewCollateral(co) [X](#), 750
 definition
 DssExecLib.allowOSMFreeze(_osm, _ilk) [X](#), 748

amt
 write
 Cure.drop(src) [X](#) [a](#), 293
 Cure.load(src) [X](#), 293
 definition
 Cure.amt: `mapping(address => uint256)`, 291

read
 Cure.drop(src) [X](#) [a](#), 293
 Cure.load(src) [X](#), 293

applyL1ToL2Alias
 call
 L2DaiGateway.modifier onlyL1Counterpart(l1Counterpart) [L2CrossDomainEnabled], 581
 L2GovernanceRelay.modifier onlyL1Counterpart(l1Counterpart)
 [L2CrossDomainEnabled], 585
 definition
 L2DaiGateway.applyL1ToL2Alias(l1Address) [L2CrossDomainEnabled], 581
 L2GovernanceRelay.applyL1ToL2Alias(l1Address) [L2CrossDomainEnabled], 585

_approvals
 write
 AAVE.approve(guy, wad) [X](#), 452
 AAVE.transferFrom(src, dst, wad) [X](#), 452
 BAL.approve(guy, wad) [X](#), 454
 BAL.transferFrom(src, dst, wad) [X](#), 454
 BAT.approve(guy, wad) [X](#), 456
 BAT.transferFrom(src, dst, wad) [X](#), 456
 COMP.approve(guy, wad) [X](#), 458
 COMP.transferFrom(src, dst, wad) [X](#), 458
 DGD.approve(guy, wad) [X](#), 460
 DGD.transferFrom(src, dst, wad) [X](#), 460
 GUSD.approve(guy, wad) [X](#), 464
 GUSD.transferFrom(src, dst, wad) [X](#), 464
 KNC.approve(guy, wad) [X](#), 466
 KNC.transferFrom(src, dst, wad) [X](#), 466
 LINK.approve(guy, wad) [X](#), 468
 LINK.transferFrom(src, dst, wad) [X](#), 468
 LRC.approve(guy, wad) [X](#), 470
 LRC.transferFrom(src, dst, wad) [X](#), 470
 MANA.approve(guy, wad) [X](#), 472
 MANA.transferFrom(src, dst, wad) [X](#), 472
 MATIC.approve(guy, wad) [X](#), 474
 MATIC.transferFrom(src, dst, wad) [X](#), 474
 OMG.approve(guy, wad) [X](#), 476
 OMG.transferFrom(src, dst, wad) [X](#), 476
 PAXG.approve(guy, wad) [X](#), 479
 PAXG.transferFrom(src, dst, wad) [X](#), 478
 PAXUSD.approve(guy, wad) [X](#), 482
 PAXUSD.transferFrom(src, dst, wad) [X](#), 482
 RENBTC.approve(guy, wad) [X](#), 484
 RENBTC.transferFrom(src, dst, wad) [X](#), 484
 REP.approve(guy, wad) [X](#), 486
 REP.transferFrom(src, dst, wad) [X](#), 486
 TUSD.approve(guy, wad) [X](#), 488
 TUSD.transferFrom(src, dst, wad) [X](#), 488
 UNI.approve(guy, wad) [X](#), 490

UNI.transferFrom(src, dst, wad) [X](#), 490
 USDC.approve(guy, wad) [X](#), 492
 USDC.transferFrom(src, dst, wad) [X](#), 492
 WBTC.approve(guy, wad) [X](#), 496
 WBTC.transferFrom(src, dst, wad) [X](#), 496
 WSTETH.approve(guy, wad) [X](#), 498
 WSTETH.transferFrom(src, dst, wad) [X](#), 498
 YFI.approve(guy, wad) [X](#), 500
 YFI.transferFrom(src, dst, wad) [X](#), 500
 ZRX.approve(guy, wad) [X](#), 502
 ZRX.transferFrom(src, dst, wad) [X](#), 502

definition

AAVE.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 451
 BAL.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 453
 BAT.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 455
 COMP.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 457
 DGD.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 459
 GUSD.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 463
 KNC.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 465
 LINK.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 467
 LRC.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 469
 MANA.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 471
 MATIC.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 473
 OMG.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 475
 PAXG.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 477
 PAXUSD.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 481
 RENBTC.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 483
 REP.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 485
 TUSD.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 487
 UNI.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 489
 USDC.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 491
 WBTC.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 495
 WSTETH.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 497
 YFI.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 499
 ZRX.approvals: [mapping\(address => mapping\(address => uint256\)\)](#), 501

read

AAVE.allowance(src, guy), 451
 AAVE.transferFrom(src, dst, wad) [X](#), 452
 BAL.allowance(src, guy), 453
 BAL.transferFrom(src, dst, wad) [X](#), 454
 BAT.allowance(src, guy), 455
 BAT.transferFrom(src, dst, wad) [X](#), 456
 COMP.allowance(src, guy), 457
 COMP.transferFrom(src, dst, wad) [X](#), 458
 DGD.allowance(src, guy), 459
 DGD.transferFrom(src, dst, wad) [X](#), 460
 GUSD.allowance(src, guy), 463
 GUSD.transferFrom(src, dst, wad) [X](#), 464
 KNC.allowance(src, guy), 465
 KNC.transferFrom(src, dst, wad) [X](#), 466
 LINK.allowance(src, guy), 467
 LINK.transferFrom(src, dst, wad) [X](#), 468
 LRC.allowance(src, guy), 469
 LRC.transferFrom(src, dst, wad) [X](#), 470
 MANA.allowance(src, guy), 471
 MANA.transferFrom(src, dst, wad) [X](#), 472
 MATIC.allowance(src, guy), 473
 MATIC.transferFrom(src, dst, wad) [X](#), 474
 OMG.allowance(src, guy), 475
 OMG.transferFrom(src, dst, wad) [X](#), 476
 PAXG.allowance(src, guy), 478
 PAXG.transferFrom(src, dst, wad) [X](#), 478
 PAXUSD.allowance(src, guy), 481

PAXUSD.transferFrom(src, dst, wad) [X](#), 482
 RENBTC.allowance(src, guy), 483
 RENBTC.transferFrom(src, dst, wad) [X](#), 484
 REP.allowance(src, guy), 485
 REP.transferFrom(src, dst, wad) [X](#), 486
 TUSD.allowance(src, guy), 487
 TUSD.transferFrom(src, dst, wad) [X](#), 488
 UNI.allowance(src, guy), 489
 UNI.transferFrom(src, dst, wad) [X](#), 490
 USDC.allowance(src, guy), 491
 USDC.transferFrom(src, dst, wad) [X](#), 492
 WBTC.allowance(src, guy), 495
 WBTC.transferFrom(src, dst, wad) [X](#), 496
 WSTETH.allowance(src, guy), 497
 WSTETH.transferFrom(src, dst, wad) [X](#), 498
 YFI.allowance(src, guy), 499
 YFI.transferFrom(src, dst, wad) [X](#), 500
 ZRX.allowance(src, guy), 501
 ZRX.transferFrom(src, dst, wad) [X](#), 502

approve

call

CalleeMakerOtcDai.clipperCall(sender, daiAmt, gemAmt, data) [X](#), 550
 CalleeMakerOtcDai.setUp(otc_, daiJoin_) [CalleeMakerOtc], 549
 CurveLpTokenUniv3Callee.constructor(uniV3Router_, daiJoin_, weth_) [X](#), 551
 CurveLpTokenUniv3Callee.clipperCall(sender, owe, slice, data) [X](#), 552
 D3MAavePool.constructor(ilk_, hub_, dai_, pool_) [X](#), 349
 D3MCompoundPool.constructor(ilk_, hub_, cDai_) [X](#), 377
 D3MHub.constructor(daiJoin_) [X](#), 320
 DAIMock.approve(spender, amount) [ERC20] [X](#), 628
 DAIMock.decreaseAllowance(spender, subtractedValue) [ERC20] [X](#), 629
 DAIMock.increaseAllowance(spender, addedValue) [ERC20] [X](#), 628
 DAIMock._spendAllowance(owner, spender, amount) [ERC20], 631
 DssProxyActions.daiJoin_join(apt, urn, wad) [Common] [X](#), 529
 DssProxyActionsDsr.daiJoin_join(apt, urn, wad) [Common] [X](#), 547
 DssProxyActionsEnd.daiJoin_join(apt, urn, wad) [Common] [X](#), 543
 DssProxyActions.ethJoin_join(apt, urn) [X](#), 531
 DssProxyActions.gemJoin_join(apt, urn, amt, transferFrom) [X](#), 531
 DssPsm.constructor(gemJoin_, daiJoin_, vow_) [X](#), 509
 L1DAITeleportGateway.constructor(_starkNet, _l1Token, _l2TeleportGateway, _l1Escrow, _l1TeleportRouter) [X](#), 623
 L1Escrow.approve(token, spender, value) [X](#) [a](#), 575, 597, 635
 L1EscrowMom.refuse(spender) [X](#) [a](#), 638
 PSMCallee.constructor(daiJoin_) [X](#), 555
 PSMCallee.clipperCall(sender, owe, slice, data) [X](#), 555
 TeleportJoin.constructor(vat_, daiJoin_, ilk_, domain_) [X](#), 641
 UniswapV2CalleeDai.clipperCall(sender, daiAmt, gemAmt, data) [X](#), 558
 UniswapV2CalleeDai.setUp(uniRouter02_, daiJoin_) [UniswapV2Callee], 557
 UniswapV2LpTokenCalleeDai.clipperCall(sender, daiAmt, gemAmt, data) [X](#), 560
 UniswapV2LpTokenCalleeDai.setUp(uniRouter02_, daiJoin_) [UniswapV2Callee(2)], 559
 UniswapV2LpTokenCalleeDai.swapGemForDai(token, path, to), 560
 UniswapV3Callee.constructor(uniV3Router_, daiJoin_) [X](#), 567
 UniswapV3Callee.clipperCall(sender, owe, slice, data) [X](#), 567
 WstETHCurveUniv3Callee.constructor(curvePool_, uniV3Router_, daiJoin_, weth_) [X](#), 563
 WstETHCurveUniv3Callee.clipperCall(sender, owe, slice, data) [X](#), 564

definition

AAVE.approve(guy, wad) [X](#), 452
 BAL.approve(guy, wad) [X](#), 454
 BAT.approve(guy, wad) [X](#), 456
 COMP.approve(guy, wad) [X](#), 458
 Dai.approve(spender, value) [X](#), 589, 615
 Dai.approve(usr, wad) [X](#), 225
 DAIMock._approve(owner, spender, amount) [ERC20], 631

DAIMock.approve(spender, amount) [ERC20] [X](#), 628
 DGD.approve(guy, wad) [X](#), 460
 GUSD.approve(guy, wad) [X](#), 464
 KNC.approve(guy, wad) [X](#), 466
 L1Escrow.approve(token, spender, value) [X](#) [a](#), 575, 597, 635
 LINK.approve(guy, wad) [X](#), 468
 LRC.approve(guy, wad) [X](#), 470
 MANA.approve(guy, wad) [X](#), 472
 MATIC.approve(guy, wad) [X](#), 474
 OMG.approve(guy, wad) [X](#), 476
 PAXG.approve(guy, wad) [X](#), 479
 PAXUSD.approve(guy, wad) [X](#), 482
 RENBTC.approve(guy, wad) [X](#), 484
 REP.approve(guy, wad) [X](#), 486
 TUSD.approve(guy, wad) [X](#), 488
 UNI.approve(guy, wad) [X](#), 490
 USDC.approve(guy, wad) [X](#), 492
 USDT.approve(_spender, _value) [X](#), 494
 WBTC.approve(guy, wad) [X](#), 496
 WETH9_.approve(guy, wad) [X](#), 517
 WSTETH.approve(guy, wad) [X](#), 498
 YFI.approve(guy, wad) [X](#), 500
 ZRX.approve(guy, wad) [X](#), 502

Art

call

D3MAavePool.exit(dst, wad) [X](#), 351
 D3MCompoundPool.exit(dst, wad) [X](#), 379

write

End.cage(ilk) [X](#), 299
 End.skip(ilk, id) [X](#), 299
 End.snip(ilk, id) [X](#), 299

definition

End.Art: [mapping\(bytes32 => uint256\)](#), 295

read

End.flow(ilk) [X](#), 300
 End.skip(ilk, id) [X](#), 299
 End.snip(ilk, id) [X](#), 299

art

write

TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643
 TeleportJoin.settle(sourceDomain, batchedDaiToFlush) [X](#), 645

definition

TeleportJoin.art: [uint256](#), 641

read

TeleportJoin.cure(), 643

Ash

call

Flopper.dent(id, lot, bid) [X](#), 267

write

Vow.cage() [X](#) [a](#), 271
 Vow.flop() [X](#), 271
 Vow.kiss(rad) [X](#), 271

definition

Vow.Ash: [uint256](#), 269

read

Vow.flap() [X](#), 271
 Vow.flop() [X](#), 271
 Vow.heal(rad) [X](#), 270
 Vow.kiss(rad) [X](#), 271

assetBalance

call

D3MAavePool.exit(dst, wad) [X](#), 351
 D3MAavePool.maxWithdraw(), 351


```

    D3MCompoundPool.maxWithdraw(), 379
    D3MHub._exec(ilk, _pool, Art, lineWad), 322
definition
    D3MAavePool.assetBalance(), 351
    D3MCompoundPool.assetBalance(), 379
assume
    call
        D3MCoreDeployScript.assumeNoPrecompiles(addr, chainId) [StdCheatsSafe], 338
        D3MCoreInitScript.assumeNoPrecompiles(addr, chainId) [StdCheatsSafe], 366
        D3MDeployScript.assumeNoPrecompiles(addr, chainId) [StdCheatsSafe], 392
        D3MInitScript.assumeNoPrecompiles(addr, chainId) [StdCheatsSafe], 412
assumeNoPrecompiles
    call
        D3MCoreDeployScript.assumeNoPrecompiles(addr) [StdCheatsSafe], 338
        D3MCoreInitScript.assumeNoPrecompiles(addr) [StdCheatsSafe], 366
        D3MDeployScript.assumeNoPrecompiles(addr) [StdCheatsSafe], 392
        D3MInitScript.assumeNoPrecompiles(addr) [StdCheatsSafe], 412
definition
    D3MCoreDeployScript.assumeNoPrecompiles(addr) [StdCheatsSafe], 338
    D3MCoreDeployScript.assumeNoPrecompiles(addr, chainId) [StdCheatsSafe], 338
    D3MCoreInitScript.assumeNoPrecompiles(addr) [StdCheatsSafe], 366
    D3MCoreInitScript.assumeNoPrecompiles(addr, chainId) [StdCheatsSafe], 366
    D3MDeployScript.assumeNoPrecompiles(addr) [StdCheatsSafe], 392
    D3MDeployScript.assumeNoPrecompiles(addr, chainId) [StdCheatsSafe], 392
    D3MInitScript.assumeNoPrecompiles(addr) [StdCheatsSafe], 412
    D3MInitScript.assumeNoPrecompiles(addr, chainId) [StdCheatsSafe], 412
.at
    call
        EnumerableSet.at(set, index), 758, 760, 761
        TeleportRouter.domainAt(index), 654
definition
    EnumerableSet._at(set, index), 757, 758, 760, 761
ATOKEN_REVISION
    call
        D3MAavePlan.active(), 314
        D3MAavePlan.constructor(dai_, pool_) X a, 311
        D3MInit.initAave(dss, d3m, cfg, aaveCfg), 726
auth
    call
        AuthGemJoin5.cage() X a, 504
        AuthGemJoin5.deny(usr) X a, 503
        AuthGemJoin5.join(urn, amt, msgSender) X a, 504
        AuthGemJoin5.rely(usr) X a, 503
        AuthGemJoin8.cage() X a, 506
        AuthGemJoin8.deny(usr) X a, 505
        AuthGemJoin8.join(urn, amt, msgSender) X a, 506
        AuthGemJoin8.rely(usr) X a, 505
        AuthGemJoin8.setImplementation(implementation, permitted) X a, 506
        AuthGemJoin.cage() X a, 447, 507
        AuthGemJoin.deny(usr) X a, 447, 507
        AuthGemJoin.join(urn, wad, msgSender) X a, 508
        AuthGemJoin.join(usr, wad) X a, 448
        AuthGemJoin.rely(usr) X a, 447, 507
        BasicRelay.addRelayers(relayers_) X a, 657
        BasicRelay.deny(usr) X a, 657
        BasicRelay.rely(usr) X a, 657
        BasicRelay.removeRelayers(relayers_) X a, 658
        ChainLog.deny(usr) X a, 709
        ChainLog.rely(usr) X a, 709
        ChainLog.removeAddress(_key) X a, 710
        ChainLog.setAddress(_key, _addr) X a, 710
        ChainLog.setIPFS(_ipfs) X a, 710
        ChainLog.setSha256sum(_sha256sum) X a, 710

```

ChainLog.setVersion(_version) [X a](#), 710
 Clipper.deny(usr) [X a](#), 242
 Clipper.file(what, [data](#)) [X a](#), 243
 Clipper.kick(tab, lot, usr, kpr) [X a](#), 244
 ClipperMom.setBreaker(clip, level, delay) [X a](#), 258
 Clipper.rely(usr) [X a](#), 242
 Clipper.yank(id) [X a](#), 249
 Cure.cage() [X a](#), 293
 Cure.deny(usr) [X a](#), 292
 Cure.drop(src) [X a](#), 293
 Cure.file(what, [data](#)) [X a](#), 292
 Cure.lift(src) [X a](#), 293
 Cure.rely(usr) [X a](#), 292
 D3MAavePlan.deny(usr) [X a](#), 312
 D3MAavePlan.file(what, [data](#)) [X a](#), 312
 D3MAavePlan.rely(usr) [X a](#), 312
 D3MAavePool.deny(usr) [X a](#), 350
 D3MAavePool.file(what, [data](#)) [X a](#), 350
 D3MAavePool.quit(dst) [X a](#), 351
 D3MAavePool.rely(usr) [X a](#), 350
 D3MCompoundPlan.deny(usr) [X a](#), 316
 D3MCompoundPlan.file(what, addr, [data](#)) [X a](#), 316
 D3MCompoundPlan.file(what, [data](#)) [X a](#), 316
 D3MCompoundPlan.rely(usr) [X a](#), 316
 D3MCompoundPool.deny(usr) [X a](#), 378
 D3MCompoundPool.file(what, [data](#)) [X a](#), 378
 D3MCompoundPool.quit(dst) [X a](#), 379
 D3MCompoundPool.rely(usr) [X a](#), 378
 D3MHub.cage(ilk) [X a](#), 325
 D3MHub.deny(usr) [X a](#), 321
 D3MHub.file(ilk, what, [data](#)) [X a](#), 321
 D3MHub.file(what, [data](#)) [X a](#), 321
 D3MHub.rely(usr) [X a](#), 321
 D3MMom.disable(who) [X a](#), 356
 D3MOracle.deny(usr) [X a](#), 353
 D3MOracle.file(what, [data](#)) [X a](#), 354
 D3MOracle.rely(usr) [X a](#), 353
 Dai.deny(guy) [X a](#), 223
 Dai.deny(usr) [X a](#), 587, 613
 DaiJoin.cage() [X a](#), 425
 DaiJoin.deny(usr) [X a](#), 425
 DaiJoin.rely(usr) [X a](#), 425
 Dai.mint(to, [value](#)) [X a](#), 590, 616
 Dai.mint(usr, wad) [X a](#), 224
 Dai.rely(guy) [X a](#), 223
 Dai.rely(usr) [X a](#), 587, 613
 Dog.cage() [X a](#), 255
 Dog.deny(usr) [X a](#), 252
 Dog.digs(ilk, rad) [X a](#), 255
 Dog.file(ilk, what, clip) [X a](#), 253
 Dog.file(ilk, what, [data](#)) [X a](#), 253
 Dog.file(what, [data](#)) [X a](#), 252
 Dog.rely(usr) [X a](#), 252
 DssAutoLine.deny(usr) [X a](#), 707
 DssAutoLine.rely(usr) [X a](#), 707
 DssAutoLine.remIlk(ilk) [X a](#), 706
 DssAutoLine.setIlk(ilk, line, gap, ttl) [X a](#), 706
 DssPsm.deny(usr) [X a](#), 509
 DssPsm.file(what, [data](#)) [X a](#), 510
 DssPsm.hope(usr) [X a](#), 510
 DssPsm.nope(usr) [X a](#), 510
 DssPsm.rely(usr) [X a](#), 509
 DssVestMintable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] [X a](#), 678

DssVestMintable.deny(_usr) [DssVest] [X a](#), 677
DssVestMintable.file(what, [data](#)) [DssVest] [X a](#), 677
DssVestMintable.rely(_usr) [DssVest] [X a](#), 677
DssVestSuckable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] [X a](#), 688
DssVestSuckable.deny(_usr) [DssVest] [X a](#), 687
DssVestSuckable.file(what, [data](#)) [DssVest] [X a](#), 687
DssVestSuckable.rely(_usr) [DssVest] [X a](#), 687
DssVestTransferrable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] [X a](#), 698
DssVestTransferrable.deny(_usr) [DssVest] [X a](#), 697
DssVestTransferrable.file(what, [data](#)) [DssVest] [X a](#), 697
DssVestTransferrable.rely(_usr) [DssVest] [X a](#), 697
End.cage() [X a](#), 299
End.deny(usr) [X a](#), 297
End.file(what, [data](#)) [X a](#), 298
End.rely(usr) [X a](#), 297
ESM.cage() [X a](#), 304
ESM.deny(usr) [X a](#), 304
ESM.file(what, [data](#)) [X a](#), 304
ESM.rely(usr) [X a](#), 304
ExponentialDecrease.deny(usr) [X a](#), 239
ExponentialDecrease.file(what, [data](#)) [X a](#), 239
ExponentialDecrease.rely(usr) [X a](#), 239
Flapper.cage(rad) [X a](#), 264
Flapper.deny(usr) [X a](#), 262
Flapper.file(what, [data](#)) [X a](#), 263
Flapper.kick(lot, bid) [X a](#), 263
Flapper.rely(usr) [X a](#), 262
Flopper.cage() [X a](#), 267
Flopper.deny(usr) [X a](#), 266
Flopper.file(what, [data](#)) [X a](#), 266
Flopper.kick(gal, lot, bid) [X a](#), 266
Flopper.rely(usr) [X a](#), 265
GemJoin2.cage() [X a](#), 427
GemJoin2.deny(usr) [X a](#), 427
GemJoin2.rely(usr) [X a](#), 427
GemJoin3.cage() [X a](#), 429
GemJoin3.deny(usr) [X a](#), 429
GemJoin3.rely(usr) [X a](#), 429
GemJoin4.cage() [X a](#), 433
GemJoin4.deny(usr) [X a](#), 433
GemJoin4.rely(usr) [X a](#), 433
GemJoin5.cage() [X a](#), 435
GemJoin5.deny(usr) [X a](#), 435
GemJoin5.rely(usr) [X a](#), 435
GemJoin6.cage() [X a](#), 438
GemJoin6.deny(usr) [X a](#), 437
GemJoin6.rely(usr) [X a](#), 437
GemJoin6.setImplementation(implementation, permitted) [X a](#), 438
GemJoin7.cage() [X a](#), 440
GemJoin7.deny(usr) [X a](#), 439
GemJoin7.rely(usr) [X a](#), 439
GemJoin7.setImplementation(implementation, permitted) [X a](#), 440
GemJoin8.cage() [X a](#), 442
GemJoin8.deny(usr) [X a](#), 441
GemJoin8.rely(usr) [X a](#), 441
GemJoin8.setImplementation(implementation, permitted) [X a](#), 442
GemJoin9.cage() [X a](#), 444
GemJoin9.deny(usr) [X a](#), 443
GemJoin9.rely(usr) [X a](#), 443
GemJoin.cage() [X a](#), 424
GemJoin.deny(usr) [X a](#), 424
GemJoin.rely(usr) [X a](#), 424
IlkRegistry.deny(usr) [X a](#), 667

IlkRegistry.file(ilk, what, data) X a, 669, 670
 IlkRegistry.file(what, data) X a, 669
 IlkRegistry.put(_ilk, _join, _gem, _dec, _class, _pip, _xlip, _name, _symbol) X a, 673
 IlkRegistry.rely(usr) X a, 667
 IlkRegistry.removeAuth(ilk) X a, 669
 Jug.deny(usr) X a, 273
 Jug.file(ilk, what, data) X a, 275
 Jug.file(what, data) X a, 275
 Jug.init(ilk) X a, 274
 Jug.rely(usr) X a, 273
 L1DAIBridge.close() X a, 618
 L1DAIBridge.deny(usr) X a, 618
 L1DAIBridge.rely(usr) X a, 618
 L1DAIBridge.setCeiling(_ceiling) X a, 619
 L1DAIBridge.setMaxDeposit(_maxDeposit) X a, 619
 L1DaiGateway.close() X a, 571
 L1DaiGateway.deny(usr) X a, 571
 L1DaiGateway.rely(usr) X a, 570
 L1DAITokenBridge.close() X a, 594
 L1DAITokenBridge.deny(usr) X a, 593
 L1DAITokenBridge.rely(usr) X a, 593
 L1Escrow.approve(token, spender, value) X a, 575, 597, 635
 L1Escrow.deny(usr) X a, 575, 597, 635
 L1EscrowMom.refuse(spender) X a, 638
 L1Escrow.rely(usr) X a, 575, 597, 635
 L1GovernanceRelay.deny(usr) X a, 578, 601, 621
 L1GovernanceRelay.reclaim(receiver, amount) X a, 579
 L1GovernanceRelay.relay(spell) X a, 621
 L1GovernanceRelay.relay(target, targetData, l1CallValue, maxGas, gasPriceBid, maxSubmissionCost) X a, 579
 L1GovernanceRelay.relay(target, targetData, l2gas) X a, 601
 L1GovernanceRelay.rely(usr) X a, 578, 601, 621
 L2DaiGateway.close() X a, 582
 L2DaiGateway.deny(usr) X a, 582
 L2DaiGateway.rely(usr) X a, 582
 L2DAITokenBridge.close() X a, 605
 L2DAITokenBridge.deny(usr) X a, 605
 L2DAITokenBridge.rely(usr) X a, 605
 LinearDecrease.deny(usr) X a, 235
 LinearDecrease.file(what, data) X a, 236
 LinearDecrease.rely(usr) X a, 235
 ManagedGemJoin.cage() X a, 450
 ManagedGemJoin.deny(usr) X a, 449
 ManagedGemJoin.exit(urn, usr, amt) X a, 450
 ManagedGemJoin.join(usr, amt) X a, 450
 ManagedGemJoin.rely(usr) X a, 449
 Median.deny(usr) X a, 282
 Median.diss(a) X a, 284
 Median.drop(a) X a, 283
 Median.kiss(a) X a, 284
 Median.lift(a) X a, 283
 Median.rely(usr) X a, 282
 Median.setBar(bar_) X a, 284
 OSM.change(src_) X a, 287
 OSM.deny(usr) X a, 286
 OSM.diss(a) X a, 288
 OSM.kiss(a) X a, 288
 OSM.rely(usr) X a, 286
 OSM.start() X a, 286
 OSM.step(ts) X a, 287
 OSM.stop() X a, 286
 OSM.void() X a, 287

Pot.cage() [X a](#), 279
 Pot.deny(guy) [X a](#), 277
 Pot.file(what, addr) [X a](#), 278
 Pot.file(what, [data](#)) [X a](#), 278
 Pot.rely(guy) [X a](#), 277
 Spotter.cage() [X a](#), 228
 Spotter.deny(guy) [X a](#), 227
 Spotter.file(ilk, what, pip_) [X a](#), 228
 Spotter.file(ilk, what, [data](#)) [X a](#), 228
 Spotter.file(what, [data](#)) [X a](#), 228
 Spotter.rely(guy) [X a](#), 227
 StairstepExponentialDecrease.deny(usr) [X a](#), 237
 StairstepExponentialDecrease.file(what, [data](#)) [X a](#), 237
 StairstepExponentialDecrease.rely(usr) [X a](#), 237
 TeleportJoin.deny(usr) [X a](#), 642
 TeleportJoin.file(what, domain_, [data](#)) [X a](#), 642
 TeleportJoin.file(what, [data](#)) [X a](#), 642
 TeleportJoin.rely(usr) [X a](#), 642
 TeleportJoin.requestMint(teleportGUID, maxFeePercentage, operatorFee) [X a](#), 644
 TeleportOracleAuth.addSigners(signers_) [X a](#), 650
 TeleportOracleAuth.deny(usr) [X a](#), 649
 TeleportOracleAuth.file(what, [data](#)) [X a](#), 649
 TeleportOracleAuth.rely(usr) [X a](#), 649
 TeleportOracleAuth.removeSigners(signers_) [X a](#), 650
 TeleportRouter.deny(usr) [X a](#), 653
 TeleportRouter.file(what, domain, [data](#)) [X a](#), 653
 TeleportRouter.rely(usr) [X a](#), 653
 TrustedRelay.addRelayers(relayers_) [X a](#), 662
 TrustedRelay.deny(usr) [X a](#), 662
 TrustedRelay.diss(usr) [X a](#), 662
 TrustedRelay.file(what, [data](#)) [X a](#), 662
 TrustedRelay.kiss(usr) [X a](#), 662
 TrustedRelay.rely(usr) [X a](#), 662
 TrustedRelay.removeRelayers(relayers_) [X a](#), 662
 Vat.cage() [X a](#), 231
 Vat.deny(usr) [X a](#), 229
 Vat.file(ilk, what, [data](#)) [X a](#), 231
 Vat.file(what, [data](#)) [X a](#), 231
 Vat.fold(i, u, rate) [X a](#), 234
 Vat.grab(i, u, v, w, dink, dart) [X a](#), 233
 Vat.init(ilk) [X a](#), 231
 Vat.rely(usr) [X a](#), 229
 Vat.slip(ilk, usr, wad) [X a](#), 231
 Vat.suck(u, v, rad) [X a](#), 234
 Vow.cage() [X a](#), 271
 Vow.deny(usr) [X a](#), 269
 Vow.fess(tab) [X a](#), 270
 Vow.file(what, [data](#)) [X a](#), 270
 Vow.rely(usr) [X a](#), 269

definition

AuthGemJoin5.modifier auth(), 503
 AuthGemJoin8.modifier auth(), 505
 AuthGemJoin.modifier auth(), 447, 507
 BasicRelay.modifier auth(), 657
 ChainLog.modifier auth(), 709
 Clipper.modifier auth(), 242
 ClipperMom.modifier auth(), 257
 Cure.modifier auth(), 291
 D3MAavePlan.modifier auth(), 311
 D3MAavePool.modifier auth(), 349
 D3MCompoundPlan.modifier auth(), 315
 D3MCompoundPool.modifier auth(), 377
 D3MHub.modifier auth(), 320

D3MMom.modifier auth(), 355
 D3MOracle.modifier auth(), 353
 Dai.modifier auth(), 223, 587, 613
 DaiJoin.modifier auth(), 425
 Dog.modifier auth(), 251
 DssAutoLine.modifier auth(), 705
 DssPsm.modifier auth(), 509
 DssVestMintable.modifier auth() [DssVest], 676
 DssVestSuckable.modifier auth() [DssVest], 686
 DssVestTransferrable.modifier auth() [DssVest], 696
 End.modifier auth(), 297
 ESM.modifier auth(), 303
 ExponentialDecrease.modifier auth(), 239
 Flapper.modifier auth(), 262
 Flopper.modifier auth(), 265
 GemJoin2.modifier auth(), 427
 GemJoin3.modifier auth(), 429
 GemJoin4.modifier auth(), 433
 GemJoin5.modifier auth(), 435
 GemJoin6.modifier auth(), 437
 GemJoin7.modifier auth(), 439
 GemJoin8.modifier auth(), 441
 GemJoin9.modifier auth(), 443
 GemJoin.modifier auth(), 423
 IlkRegistry.modifier auth(), 667
 Jug.modifier auth(), 273
 L1DAIBridge.modifier auth(), 618
 L1DaiGateway.modifier auth(), 569
 L1DAITokenBridge.modifier auth(), 592
 L1Escrow.modifier auth(), 575, 597, 635
 L1EscrowMom.modifier auth(), 637
 L1GovernanceRelay.modifier auth(), 577, 600, 621
 L2DaiGateway.modifier auth(), 581
 L2DAITokenBridge.modifier auth(), 604
 LinearDecrease.modifier auth(), 235
 ManagedGemJoin.modifier auth(), 449
 Median.modifier auth(), 282
 OSM.modifier auth(), 286
 Pot.modifier auth(), 277
 Spotter.modifier auth(), 227
 StairstepExponentialDecrease.modifier auth(), 237
 TeleportJoin.modifier auth(), 641
 TeleportOracleAuth.modifier auth(), 649
 TeleportRouter.modifier auth(), 653
 TrustedRelay.modifier auth(), 661
 Vat.modifier auth(), 229
 Vow.modifier auth(), 269

AuthGemJoin

definition

contract AuthGemJoin, 447
 contract AuthGemJoin(2), 507

AuthGemJoin5

definition

contract AuthGemJoin5, 503

AuthGemJoin8

definition

contract AuthGemJoin8, 505

authority

write

ClipperMom.setAuthority(authority_) X, 258
 D3MMom.setAuthority(authority_) X, 356
 L1EscrowMom.setAuthority(authority_) X, 638

definition

```

ClipperMom.authority: address, 257
D3MMom.authority: address, 355
L1EscrowMom.authority: address, 637
read
ClipperMom.isAuthorized(src, sig), 258
ClipperMom.setAuthority(authority_) X, 258
D3MMom.isAuthorized(src, sig), 355
L1EscrowMom.isAuthorized(src, sig), 637
L1EscrowMom.setAuthority(authority_) X, 638
authorize
call
DssExecLib.addCollateralBase(_ilk, _gem, _join, _clip, _calc, _pip) X, 749
DssExecLib.addNewCollateral(co) X, 750
definition
DssExecLib.authorize(_base, _ward) X, 733
autoLine
call
DssExecLib.removeIlkFromAutoLine(_ilk) X, 743
DssExecLib.setIlkAutoLineDebtCeiling(_ilk, _amount) X, 743
DssExecLib.setIlkAutoLineParameters(_ilk, _amount, _gap, _ttl) X, 742
definition
DssExecLib.autoLine(), 732
Award
definition
struct DssVest.Award, 675, 685, 695
awards
write
DssVestMintable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a, 678
DssVestMintable.move(_id, _dst) [DssVest] X, 682
DssVestMintable.restrict(_id) [DssVest] X, 681
DssVestMintable.unrestrict(_id) [DssVest] X, 681
DssVestMintable._vest(_id, _maxAmt) [DssVest], 679
DssVestMintable._yank(_id, _end) [DssVest], 682
DssVestSuckable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a, 688
DssVestSuckable.move(_id, _dst) [DssVest] X, 692
DssVestSuckable.restrict(_id) [DssVest] X, 690
DssVestSuckable.unrestrict(_id) [DssVest] X, 691
DssVestSuckable._vest(_id, _maxAmt) [DssVest], 689
DssVestSuckable._yank(_id, _end) [DssVest], 691
DssVestTransferrable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a, 698
DssVestTransferrable.move(_id, _dst) [DssVest] X, 702
DssVestTransferrable.restrict(_id) [DssVest] X, 701
DssVestTransferrable.unrestrict(_id) [DssVest] X, 701
DssVestTransferrable._vest(_id, _maxAmt) [DssVest], 699
DssVestTransferrable._yank(_id, _end) [DssVest], 701
definition
DssVestMintable.awards [DssVest] : mapping(uint256 => DssVest.Award), 675
DssVestSuckable.awards [DssVest] : mapping(uint256 => DssVest.Award), 685
DssVestTransferrable.awards [DssVest] : mapping(uint256 => DssVest.Award), 695
read
DssVestMintable.accrued(_id) [DssVest], 680
DssVestMintable.bgn(_id) [DssVest], 676
DssVestMintable.clf(_id) [DssVest], 676
DssVestMintable.fin(_id) [DssVest], 676
DssVestMintable.mgr(_id) [DssVest], 676
DssVestMintable.move(_id, _dst) [DssVest] X, 682
DssVestMintable.res(_id) [DssVest], 677
DssVestMintable.restrict(_id) [DssVest] X, 681
DssVestMintable.rxd(_id) [DssVest], 677
DssVestMintable.tot(_id) [DssVest], 677
DssVestMintable.unpaid(_id) [DssVest], 680
DssVestMintable.unrestrict(_id) [DssVest] X, 681
DssVestMintable.usr(_id) [DssVest], 676

```

DssVestMintable.valid(_id) [DssVest], 682
 DssVestMintable._vest(_id, _maxAmt) [DssVest], 679
 DssVestMintable._yank(_id, _end) [DssVest], 682
 DssVestSuckable.accrued(_id) [DssVest], 689
 DssVestSuckable.bgn(_id) [DssVest], 686
 DssVestSuckable.clf(_id) [DssVest], 686
 DssVestSuckable.fin(_id) [DssVest], 686
 DssVestSuckable.mgr(_id) [DssVest], 686
 DssVestSuckable.move(_id, _dst) [DssVest] **X**, 692
 DssVestSuckable.res(_id) [DssVest], 686
 DssVestSuckable.restrict(_id) [DssVest] **X**, 690
 DssVestSuckable.rxd(_id) [DssVest], 687
 DssVestSuckable.tot(_id) [DssVest], 686
 DssVestSuckable.unpaid(_id) [DssVest], 690
 DssVestSuckable.unrestrict(_id) [DssVest] **X**, 691
 DssVestSuckable.usr(_id) [DssVest], 686
 DssVestSuckable.valid(_id) [DssVest], 692
 DssVestSuckable._vest(_id, _maxAmt) [DssVest], 689
 DssVestSuckable._yank(_id, _end) [DssVest], 691
 DssVestTransferrable.accrued(_id) [DssVest], 700
 DssVestTransferrable.bgn(_id) [DssVest], 696
 DssVestTransferrable.clf(_id) [DssVest], 696
 DssVestTransferrable.fin(_id) [DssVest], 696
 DssVestTransferrable.mgr(_id) [DssVest], 696
 DssVestTransferrable.move(_id, _dst) [DssVest] **X**, 702
 DssVestTransferrable.res(_id) [DssVest], 696
 DssVestTransferrable.restrict(_id) [DssVest] **X**, 701
 DssVestTransferrable.rxd(_id) [DssVest], 697
 DssVestTransferrable.tot(_id) [DssVest], 697
 DssVestTransferrable.unpaid(_id) [DssVest], 700
 DssVestTransferrable.unrestrict(_id) [DssVest] **X**, 701
 DssVestTransferrable.usr(_id) [DssVest], 696
 DssVestTransferrable.valid(_id) [DssVest], 702
 DssVestTransferrable._vest(_id, _maxAmt) [DssVest], 699
 DssVestTransferrable._yank(_id, _end) [DssVest], 701

B

bag

write

End.pack(wad) **X**, 301

definition

End.bag: **mapping**(address => uint256), 295

read

End.cash(ilk, wad) **X**, 301

End.pack(wad) **X**, 301

bags

call

DssProxyActions.openLockGNTAndDraw(manager, jug, gntJoin, daiJoin, ilk, amtC, wadD) **X**, 539

write

GemJoin4.make(usr) **X**, 434

definition

GemJoin4.bags: **mapping**(address => address), 433

read

GemJoin4.join(usr, wad) **X**, 434

GemJoin4.make(usr) **X**, 434

BAL

definition

contract BAL, 453

balanceOf

call

CalleeMakerOtcDai.clipperCall(sender, daiAmt, gemAmt, data) **X**, 550

CurveLpTokenUniv3Callee.clipperCall(sender, owe, slice, data) **X**, 552

D3MAavePlan.getTargetAssets(currentAssets), 313

D3MAavePool.assetBalance(), 351

D3MAavePool.maxWithdraw(), 351

D3MAavePool.quit(dst) **X** a, 351

D3MAavePool.withdraw(wad) **X**, 351

D3MCompoundPool.collect(claim) **X**, 380

D3MCompoundPool.deposit(wad) **X**, 378

D3MCompoundPool.exit(dst, wad) **X**, 379

D3MCompoundPool.quit(dst) **X** a, 379

D3MCompoundPool.withdraw(wad) **X**, 379

ESM.burn() **X**, 305

GemJoin2.exit(usr, wad) **X**, 428

GemJoin2.join(usr, wad) **X**, 428

GemJoin7.join(usr, amt) **X**, 440

GemJoin9._join(usr), 444

GodMode.setBalance(token, who, amount), 768

L1DAIBridge.deposit(amount, l2Recipient) **X**, 619

PSMCallee.clipperCall(sender, owe, slice, data) **X**, 555

UniswapV2CalleeDai.clipperCall(sender, daiAmt, gemAmt, data) **X**, 558

UniswapV2LpTokenCalleeDai.clipperCall(sender, daiAmt, gemAmt, data) **X**, 560

UniswapV2LpTokenCalleeDai.swapGemForDai(token, path, to), 560

UniswapV3Callee.clipperCall(sender, owe, slice, data) **X**, 567

WstETHCurveUniv3Callee.clipperCall(sender, owe, slice, data) **X**, 564

write

Dai.transfer(to, value) **X**, 588, 614

Dai.burn(from, value) **X**, 590, 616

Dai.burn(usr, wad) **X**, 225

Dai.mint(to, value) **X** a, 590, 616

Dai.mint(usr, wad) **X** a, 224

Dai.transferFrom(from, to, value) **X**, 589, 615

Dai.transferFrom(src, dst, wad) **X**, 224

WETH9_.deposit() **X**, 517

WETH9_.transferFrom(src, dst, wad) **X**, 518

WETH9_.withdraw(wad) **X**, 517

definition

AAVE.balanceOf(src), 451

B

BAL.balanceOf(src), 453
 BAT.balanceOf(src), 455
 COMP.balanceOf(src), 457
 Dai.balanceOf: `mapping(address => uint256)`, 223, 587, 613
 DAIMock.balanceOf(account) [ERC20], 627
 DGD.balanceOf(src), 459
 GNT.balanceOf(src), 461
 GUSD.balanceOf(src), 463
 KNC.balanceOf(src), 465
 LINK.balanceOf(src), 467
 LRC.balanceOf(src), 469
 MANA.balanceOf(src), 471
 MATIC.balanceOf(src), 473
 OMG.balanceOf(src), 475
 PAXG.balanceOf(src), 478
 PAXUSD.balanceOf(src), 481
 RENBTC.balanceOf(src), 483
 REP.balanceOf(src), 485
 TUSD.balanceOf(src), 487
 UNI.balanceOf(src), 489
 USDC.balanceOf(src), 491
 USDT.balanceOf(_owner), 493
 WBTC.balanceOf(src), 495
 WETH9_.balanceOf: `mapping(address => uint256)`, 517
 WSTETH.balanceOf(src), 497
 YFI.balanceOf(src), 499
 ZRX.balanceOf(src), 501

read

Dai.transfer(to, value) X, 588, 614
 Dai.burn(from, value) X, 590, 616
 Dai.burn(usr, wad) X, 225
 Dai.mint(to, value) X a, 590, 616
 Dai.mint(usr, wad) X a, 224
 Dai.transferFrom(from, to, value) X, 589, 615
 Dai.transferFrom(src, dst, wad) X, 224
 WETH9_.deposit() X, 517
 WETH9_.transferFrom(src, dst, wad) X, 518
 WETH9_.withdraw(wad) X, 517

_balances

write

AAVE.constructor(supply) X, 451
 AAVE.transferFrom(src, dst, wad) X, 452
 BAL.constructor(supply) X, 453
 BAL.transferFrom(src, dst, wad) X, 454
 BAT.constructor(supply) X, 455
 BAT.transferFrom(src, dst, wad) X, 456
 COMP.constructor(supply) X, 457
 COMP.transferFrom(src, dst, wad) X, 458
 DAIMock._burn(account, amount) [ERC20], 630
 DAIMock._mint(account, amount) [ERC20], 630
 DAIMock._transfer(from, to, amount) [ERC20], 629
 DGD.constructor(supply) X, 459
 DGD.transferFrom(src, dst, wad) X, 460
 GNT.constructor(supply) X, 461
 GNT.transfer(dst, wad) X, 461
 GUSD.constructor(supply) X, 463
 GUSD.transferFrom(src, dst, wad) X, 464
 KNC.constructor(supply) X, 465
 KNC.transferFrom(src, dst, wad) X, 466
 LINK.constructor(supply) X, 467
 LINK.transferFrom(src, dst, wad) X, 468
 LRC.constructor(supply) X, 469
 LRC.transferFrom(src, dst, wad) X, 470

MANA.[constructor](#)(supply) [X](#), 471
 MANA.[transferFrom](#)(src, dst, wad) [X](#), 472
 MATIC.[constructor](#)(supply) [X](#), 473
 MATIC.[transferFrom](#)(src, dst, wad) [X](#), 474
 OMG.[constructor](#)(supply) [X](#), 475
 OMG.[transferFrom](#)(src, dst, wad) [X](#), 476
 PAXG.[constructor](#)(supply, fee) [X](#), 478
 PAXG.[transferFrom](#)(src, dst, wad) [X](#), 478
 PAXUSD.[constructor](#)(supply) [X](#), 481
 PAXUSD.[transferFrom](#)(src, dst, wad) [X](#), 482
 RENBTC.[constructor](#)(supply) [X](#), 483
 RENBTC.[transferFrom](#)(src, dst, wad) [X](#), 484
 REP.[constructor](#)(supply) [X](#), 485
 REP.[transferFrom](#)(src, dst, wad) [X](#), 486
 TUSD.[constructor](#)(supply) [X](#), 487
 TUSD.[transferFrom](#)(src, dst, wad) [X](#), 488
 UNI.[constructor](#)(supply) [X](#), 489
 UNI.[transferFrom](#)(src, dst, wad) [X](#), 490
 USDC.[constructor](#)(supply) [X](#), 491
 USDC.[transferFrom](#)(src, dst, wad) [X](#), 492
 USDT.[constructor](#)(initialSupply) [X](#), 493
 USDT.[transfer](#)(to, value) [X](#), 493
 USDT.[transferFrom](#)(from, to, value) [X](#), 494
 WBTC.[constructor](#)(supply) [X](#), 495
 WBTC.[transferFrom](#)(src, dst, wad) [X](#), 496
 WSTETH.[constructor](#)(supply) [X](#), 497
 WSTETH.[transferFrom](#)(src, dst, wad) [X](#), 498
 YFI.[constructor](#)(supply) [X](#), 499
 YFI.[transferFrom](#)(src, dst, wad) [X](#), 500
 ZRX.[constructor](#)(supply) [X](#), 501
 ZRX.[transferFrom](#)(src, dst, wad) [X](#), 502

definition

AAVE.balances: [mapping](#)(address => uint256), 451
 BAL.balances: [mapping](#)(address => uint256), 453
 BAT.balances: [mapping](#)(address => uint256), 455
 COMP.balances: [mapping](#)(address => uint256), 457
 DAIMock.balances [ERC20] : [mapping](#)(address => uint256), 625
 DGD.balances: [mapping](#)(address => uint256), 459
 GNT.balances: [mapping](#)(address => uint256), 461
 GUSD.balances: [mapping](#)(address => uint256), 463
 KNC.balances: [mapping](#)(address => uint256), 465
 LINK.balances: [mapping](#)(address => uint256), 467
 LRC.balances: [mapping](#)(address => uint256), 469
 MANA.balances: [mapping](#)(address => uint256), 471
 MATIC.balances: [mapping](#)(address => uint256), 473
 OMG.balances: [mapping](#)(address => uint256), 475
 PAXG.balances: [mapping](#)(address => uint256), 477
 PAXUSD.balances: [mapping](#)(address => uint256), 481
 RENBTC.balances: [mapping](#)(address => uint256), 483
 REP.balances: [mapping](#)(address => uint256), 485
 TUSD.balances: [mapping](#)(address => uint256), 487
 UNI.balances: [mapping](#)(address => uint256), 489
 USDC.balances: [mapping](#)(address => uint256), 491
 USDT.balances: [mapping](#)(address => uint256), 493
 WBTC.balances: [mapping](#)(address => uint256), 495
 WSTETH.balances: [mapping](#)(address => uint256), 497
 YFI.balances: [mapping](#)(address => uint256), 499
 ZRX.balances: [mapping](#)(address => uint256), 501

read

AAVE.balanceOf(src), 451
 AAVE.[transferFrom](#)(src, dst, wad) [X](#), 452
 BAL.balanceOf(src), 453
 BAL.[transferFrom](#)(src, dst, wad) [X](#), 454

B

BAT.balanceOf(src), 455
 BAT.transferFrom(src, dst, wad) X, 456
 COMP.balanceOf(src), 457
 COMP.transferFrom(src, dst, wad) X, 458
 DAIMock.balanceOf(account) [ERC20], 627
 DAIMock.burn(account, amount) [ERC20], 630
 DAIMock.mint(account, amount) [ERC20], 630
 DAIMock.transfer(from, to, amount) [ERC20], 629
 DGD.balanceOf(src), 459
 DGD.transferFrom(src, dst, wad) X, 460
 GNT.balanceOf(src), 461
 GNT.transfer(dst, wad) X, 461
 GUSD.balanceOf(src), 463
 GUSD.transferFrom(src, dst, wad) X, 464
 KNC.balanceOf(src), 465
 KNC.transferFrom(src, dst, wad) X, 466
 LINK.balanceOf(src), 467
 LINK.transferFrom(src, dst, wad) X, 468
 LRC.balanceOf(src), 469
 LRC.transferFrom(src, dst, wad) X, 470
 MANA.balanceOf(src), 471
 MANA.transferFrom(src, dst, wad) X, 472
 MATIC.balanceOf(src), 473
 MATIC.transferFrom(src, dst, wad) X, 474
 OMG.balanceOf(src), 475
 OMG.transferFrom(src, dst, wad) X, 476
 PAXG.balanceOf(src), 478
 PAXG.transferFrom(src, dst, wad) X, 478
 PAXUSD.balanceOf(src), 481
 PAXUSD.transferFrom(src, dst, wad) X, 482
 RENBTC.balanceOf(src), 483
 RENBTC.transferFrom(src, dst, wad) X, 484
 REP.balanceOf(src), 485
 REP.transferFrom(src, dst, wad) X, 486
 TUSD.balanceOf(src), 487
 TUSD.transferFrom(src, dst, wad) X, 488
 UNI.balanceOf(src), 489
 UNI.transferFrom(src, dst, wad) X, 490
 USDC.balanceOf(src), 491
 USDC.transferFrom(src, dst, wad) X, 492
 USDT.balanceOf(_owner), 493
 USDT.transfer(_to, _value) X, 493
 USDT.transferFrom(_from, _to, _value) X, 494
 WBTC.balanceOf(src), 495
 WBTC.transferFrom(src, dst, wad) X, 496
 WSTETH.balanceOf(src), 497
 WSTETH.transferFrom(src, dst, wad) X, 498
 YFI.balanceOf(src), 499
 YFI.transferFrom(src, dst, wad) X, 500
 ZRX.balanceOf(src), 501
 ZRX.transferFrom(src, dst, wad) X, 502

bar

write

D3MAavePlan.disable() X, 314
 D3MAavePlan.file(what, data) X a, 312
 Median.setBar(bar_) X a, 284

definition

D3MAavePlan.bar: uint256, 311
 Median.bar: uint256, 281

read

D3MAavePlan.active(), 314
 D3MAavePlan.getTargetAssets(currentAssets), 313
 Median.poke(val_, age_, v, r, s) X, 283

barb

write

D3MCompoundPlan.disable() [X](#), 317

D3MCompoundPlan.file(what, [data](#)) [X](#) [a](#), 316

definition

D3MCompoundPlan.barb: [uint256](#), 315

read

D3MCompoundPlan.active(), 317

D3MCompoundPlan.getTargetAssets(currentAssets), 317

bark

definition

Dog.bark(ilk, urn, kpr) [X](#), 253

base

write

Jug.file(what, [data](#)) [X](#) [a](#), 275

definition

Jug.base: [uint256](#), 273

read

Jug.drip(ilk) [X](#), 275

baseRatePerBlock

call

D3MCompoundPlan._calculateTargetSupply(targetInterestRate, borrows), 316

baseVariableBorrowRate

call

D3MAavePlan._calculateTargetSupply(targetInterestRate, totalDebt), 313

BasicRelay

definition

[contract](#) BasicRelay, 657

basisPointsRate

write

USDT.changeFees(_basisPointsRate, _maximumFee) [X](#), 493

definition

USDT.basisPointsRate: [uint256](#), 493

read

USDT.[transfer](#)(_to, _value) [X](#), 493

USDT.transferFrom(_from, _to, _value) [X](#), 494

BAT

definition

[contract](#) BAT, 455

_beforeTokenTransfer

call

DAIMock._burn(account, amount) [ERC20], 630

DAIMock._mint(account, amount) [ERC20], 630

DAIMock._transfer(from, to, amount) [ERC20], 629

beg

write

Flapper.file(what, [data](#)) [X](#) [a](#), 263

Flopper.file(what, [data](#)) [X](#) [a](#), 266

definition

Flapper.beg: [uint256](#), 261

Flopper.beg: [uint256](#), 265

read

Flapper.tend(id, lot, bid) [X](#), 263

Flopper.dent(id, lot, bid) [X](#), 267

bgn

write

DssVestMintable._yank(_id, _end) [DssVest], 682

DssVestSuckable._yank(_id, _end) [DssVest], 691

DssVestTransferrable._yank(_id, _end) [DssVest], 701

definition

DssVestMintable.bgn(_id) [DssVest], 676

DssVestSuckable.bgn(_id) [DssVest], 686

DssVestTransferrable.bgn(_id) [DssVest], 696

B

read

- DssVestMintable.bgn(_id) [DssVest], 676
- DssVestSuckable.bgn(_id) [DssVest], 686
- DssVestTransferrable.bgn(_id) [DssVest], 696

Bid

definition

- struct Flapper.Bid, 261
- struct Flopper.Bid, 265

bid

write

- Flapper.kick(lot, bid) X a, 263
- Flapper.tend(id, lot, bid) X, 263
- Flopper.kick(gal, lot, bid) X a, 266

read

- Flapper.deal(id) X, 264
- Flapper.tend(id, lot, bid) X, 263
- Flapper.yank(id) X, 264
- Flopper.dent(id, lot, bid) X, 267
- Flopper.yank(id) X, 267

bids

call

- End.skip(ilk, id) X, 299

write

- Flapper.deal(id) X, 264
- Flapper.kick(lot, bid) X a, 263
- Flapper.tend(id, lot, bid) X, 263
- Flapper.tick(id) X, 263
- Flapper.yank(id) X, 264
- Flopper.deal(id) X, 267
- Flopper.dent(id, lot, bid) X, 267
- Flopper.kick(gal, lot, bid) X a, 266
- Flopper.tick(id) X, 267
- Flopper.yank(id) X, 267

definition

- Flapper.bids: mapping(uint256 => Flapper.Bid), 261
- Flopper.bids: mapping(uint256 => Flopper.Bid), 265

read

- Flapper.deal(id) X, 264
- Flapper.kick(lot, bid) X a, 263
- Flapper.tend(id, lot, bid) X, 263
- Flapper.tick(id) X, 263
- Flapper.yank(id) X, 264
- Flopper.deal(id) X, 267
- Flopper.dent(id, lot, bid) X, 267
- Flopper.kick(gal, lot, bid) X a, 266
- Flopper.tick(id) X, 267
- Flopper.yank(id) X, 267

blessed

write

- TeleportJoin.requestMint(teleportGUID, maxFeePercentage, operatorFee) X a, 644

read

- TeleportJoin.requestMint(teleportGUID, maxFeePercentage, operatorFee) X a, 644

BLN

definition

- Clipper.BLN: uint256, 241

read

- Clipper.getFeedPrice(), 244

blockHash

write

- D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341
- D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369
- D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395
- D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415

read
 D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341
 D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369
 D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395
 D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415
 blockNumber
 write
 D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341
 D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369
 D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395
 D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415
 read
 D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341
 D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369
 D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395
 D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415
 both
 call
 Vat.fork(ilk, src, dst, dink, dart) **X**, 233
 Vat.frob(i, u, v, w, dink, dart) **X**, 232
 definition
 Vat.both(x, y), 232
 _bound
 call
 D3MCoreDeployScript.bound(x, min, max) [StdUtils], 336
 D3MCoreInitScript.bound(x, min, max) [StdUtils], 364
 D3MDeployScript.bound(x, min, max) [StdUtils], 390
 D3MInitScript.bound(x, min, max) [StdUtils], 410
 definition
 D3MCoreDeployScript._bound(x, min, max) [StdUtils], 335, 336
 D3MCoreInitScript._bound(x, min, max) [StdUtils], 363, 364
 D3MDeployScript._bound(x, min, max) [StdUtils], 389, 390
 D3MInitScript._bound(x, min, max) [StdUtils], 409, 410
 BPS
 definition
 D3MInitScript.BPS: **uint256**, 403
 read
 D3MInitScript.run() **X**, 420
 BPS_ONE_HUNDRED_PCT
 definition
 DssExecLib.BPS_ONE_HUNDRED_PCT: **uint256**, 729
 read
 DssExecLib.setAuctionPermittedDrop(_ilk, _pct_bps) **X**, 745
 DssExecLib.setD3MTargetInterestRate(_d3m, _pct_bps) **X**, 749
 DssExecLib.setDebtAuctionMKRIncreaseRate(_pct_bps) **X**, 740
 DssExecLib.setExponentialDecrease(_calc, _pct_bps) **X**, 747
 DssExecLib.setIlkLiquidationPenalty(_ilk, _pct_bps) **X**, 743
 DssExecLib.setIlkLiquidationRatio(_ilk, _pct_bps) **X**, 744
 DssExecLib.setKeeperIncentivePercent(_ilk, _pct_bps) **X**, 745
 DssExecLib.setLiquidationBreakerPriceTolerance(_clip, _pct_bps) **X**, 745
 DssExecLib.setMinDebtAuctionBidIncrease(_pct_bps) **X**, 739
 DssExecLib.setMinSurplusAuctionBidIncrease(_pct_bps) **X**, 738
 DssExecLib.setStairstepExponentialDecrease(_calc, _duration, _pct_bps) **X**, 746
 DssExecLib.setStartingPriceMultiplicativeFactor(_ilk, _pct_bps) **X**, 744
 BPS_ONE_PCT
 definition
 DssExecLib.BPS_ONE_PCT: **uint256**, 729
 bridge
 call
 L1DaiGateway.modifier onlyL2Counterpart(l2Counterpart) [L1CrossDomainEnabled], 569
 L1GovernanceRelay.modifier onlyL2Counterpart(l2Counterpart)
 [L1CrossDomainEnabled], 577
 bud

```

write
  Median.diss(a) X a, 284
  Median.kiss(a) X a, 284
  OSM.diss(a) X a, 288
  OSM.kiss(a) X a, 288
definition
  Median.bud: mapping(address => uint256), 281
  OSM.bud: mapping(address => uint256), 285
read
  Median.modifier toll(), 282
  OSM.modifier toll(), 286
buds
  write
    TrustedRelay.diss(usr) X a, 662
    TrustedRelay.kiss(usr) X a, 662
  definition
    TrustedRelay.buds: mapping(address => uint256), 661
  read
    TrustedRelay.modifier toll(), 661
buf
  write
    Clipper.constructor(vat_, spotter_, dog_, ilk_) X, 243
    Clipper.file(what, data) X a, 243
  definition
    Clipper.buf: uint256, 241
  read
    Clipper.kick(tab, lot, usr, kpr) X a, 244
    Clipper.redo(id, kpr) X, 245
build
  call
    DssProxyActions.giveToProxy(proxyRegistry, manager, cdp, dst) X, 532
bump
  call
    DssExecLib.setRWAILkDebtCeiling(_ilk, _ceiling, _price) X, 742
  write
    Vow.file(what, data) X a, 270
  definition
    Vow.bump: uint256, 269
  read
    Vow.flap() X, 271
_burn
  call
    DaiJoin.join(usr, wad) X, 426
    ESM.burn() X, 305
    Flapper.deal(id) X, 264
    L2DaiGateway.outboundTransfer(l1Token, to, amount, , , data) X, 583
    L2DAITokenBridge._initiateWithdrawal(_from, _to, _amount, _l1Gas, _data), 606
  definition
    Dai.burn(from, value) X, 590, 616
    Dai.burn(usr, wad) X, 225
    DAIMock._burn(account, amount) [ERC20], 630
    ESM.burn() X, 305
buyGem
  definition
    DssPsm.buyGem(usr, gemAmt) X, 511
Bytes32Set
  definition
    struct EnumerableSet.Bytes32Set, 755
bytes32ToAddress
  call
    BasicRelay.relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v,
      r, s) X, 658
    TeleportJoin._mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643

```

TeleportJoin.mintPending(teleportGUID, maxFeePercentage, operatorFee) **X**, 644
TeleportOracleAuth.requestMint(teleportGUID, signatures, maxFeePercentage, operatorFee) **X**, 650
TrustedRelay.requestMint(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s), 664
bytes32ToStr
 call
 IlkRegistry.add(adapter) **X**, 668
 definition
 IlkRegistry.bytes32ToStr(_bytes32), 674
bytesToBytes32
 call
 MCD.getIlk(dss, gem, variant), 720
 definition
 MCD.bytesToBytes32(b), 719
_bytesToUint
 call
 D3MCoreDeployScript.rawToConvertedEIP1559Detail(rawDetail) [StdCheatsSafe], 339
 D3MCoreDeployScript.rawToConvertedReceipt(rawReceipt) [StdCheatsSafe], 341
 D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341
 D3MCoreInitScript.rawToConvertedEIP1559Detail(rawDetail) [StdCheatsSafe], 367
 D3MCoreInitScript.rawToConvertedReceipt(rawReceipt) [StdCheatsSafe], 369
 D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369
 D3MDeployScript.rawToConvertedEIP1559Detail(rawDetail) [StdCheatsSafe], 393
 D3MDeployScript.rawToConvertedReceipt(rawReceipt) [StdCheatsSafe], 395
 D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395
 D3MInitScript.rawToConvertedEIP1559Detail(rawDetail) [StdCheatsSafe], 414
 D3MInitScript.rawToConvertedReceipt(rawReceipt) [StdCheatsSafe], 415
 D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415
 definition
 D3MCoreDeployScript._bytesToUint(b) [StdCheatsSafe], 343
 D3MCoreDeployScript.bytesToUint(b) [StdUtils], 337
 D3MCoreInitScript._bytesToUint(b) [StdCheatsSafe], 371
 D3MCoreInitScript.bytesToUint(b) [StdUtils], 365
 D3MDeployScript._bytesToUint(b) [StdCheatsSafe], 397
 D3MDeployScript.bytesToUint(b) [StdUtils], 391
 D3MInitScript._bytesToUint(b) [StdCheatsSafe], 417
 D3MInitScript.bytesToUint(b) [StdUtils], 411

C

cage

call

End.cage() [X](#) [a](#), 299
 ESM.fire() [X](#), 304
 Vow.cage() [X](#) [a](#), 271

definition

AuthGemJoin5.cage() [X](#) [a](#), 504
 AuthGemJoin8.cage() [X](#) [a](#), 506
 AuthGemJoin.cage() [X](#) [a](#), 447, 507
 Cure.cage() [X](#) [a](#), 293
 D3MHub.cage(ilk) [X](#) [a](#), 325
 DaiJoin.cage() [X](#) [a](#), 425
 Dog.cage() [X](#) [a](#), 255
 End.cage() [X](#) [a](#), 299
 End.cage(ilk) [X](#), 299
 ESM.cage() [X](#) [a](#), 304
 Flapper.cage(rad) [X](#) [a](#), 264
 Flopper.cage() [X](#) [a](#), 267
 GemJoin2.cage() [X](#) [a](#), 427
 GemJoin3.cage() [X](#) [a](#), 429
 GemJoin4.cage() [X](#) [a](#), 433
 GemJoin5.cage() [X](#) [a](#), 435
 GemJoin6.cage() [X](#) [a](#), 438
 GemJoin7.cage() [X](#) [a](#), 440
 GemJoin8.cage() [X](#) [a](#), 442
 GemJoin9.cage() [X](#) [a](#), 444
 GemJoin.cage() [X](#) [a](#), 424
 ManagedGemJoin.cage() [X](#) [a](#), 450
 Pot.cage() [X](#) [a](#), 279
 Spotter.cage() [X](#) [a](#), 228
 Vat.cage() [X](#) [a](#), 231
 Vow.cage() [X](#) [a](#), 271

calc

call

DssExecLib.calc(_ilk), 732

write

Clipper.file(what, [data](#)) [X](#) [a](#), 243

definition

Clipper.calc: AbacusLike, 241
 DssExecLib.calc(_ilk), 732

read

Clipper.status(tic, top), 248

_calculateDomainSeparator

call

Dai.constructor() [X](#), 588, 614
 Dai.DOMAIN_SEPARATOR(), 588, 614
 Dai.permit(owner, spender, [value](#), deadline, v, r, s) [X](#), 590, 616

definition

Dai._calculateDomainSeparator(chainId), 588, 614

calculateL2TokenAddress

definition

L1DaiGateway.calculateL2TokenAddress(l1Token), 573
 L2DaiGateway.calculateL2TokenAddress(l1Token), 584

_calculateTargetSupply

call

D3MAavePlan.getTargetAssets(currentAssets), 313
 D3MCompoundPlan.getTargetAssets(currentAssets), 317

definition

D3MAavePlan._calculateTargetSupply(targetInterestRate, totalDebt), 313
 D3MCompoundPlan._calculateTargetSupply(targetInterestRate, borrows), 316

CalleeMakerOtcDai

definition


```

    contract CalleeMakerOtcDai, 549
can
  call
    DssProxyActions.draw(manager, jug, daiJoin, cdp, wad) X, 536
    DssProxyActionsDsr.exit(daiJoin, pot, wad) X, 548
    DssProxyActionsDsr.exitAll(daiJoin, pot) X, 548
    DssProxyActionsDsr.join(daiJoin, pot, wad) X, 547
    DssProxyActionsEnd.free(manager, end, cdp), 543
    DssProxyActionsEnd.pack(daiJoin, end, wad) X, 544
    DssProxyActions.lockETHAndDraw(manager, jug, ethJoin, daiJoin, cdp, wadD) X, 538
    DssProxyActions.lockGemAndDraw(manager, jug, gemJoin, daiJoin, cdp, amtC, wadD,
      transferFrom) X, 539
  write
    Vat.hope(usr) X, 230
    Vat.nope(usr) X, 230
  definition
    Vat.can: mapping(address => mapping(address => uint256)), 229
  read
    Vat.wish(bit, usr), 230
canCall
  call
    ClipperMom.isAuthorized(src, sig), 258
    D3MMom.isAuthorized(src, sig), 355
    L1EscrowMom.isAuthorized(src, sig), 637
canCast
  call
    DssAction.modifier limited(), 307
  definition
    DssExecLib.canCast(_ts, _officeHours), 734
cancelDeposit
  definition
    L1DAIBridge.cancelDeposit(amount, l2Recipient, l1Recipient, nonce) X, 620
cancelL1ToL2Message
  call
    L1DAIBridge.cancelDeposit(amount, l2Recipient, l1Recipient, nonce) X, 620
cap
  write
    DssVestMintable.file(what, data) [DssVest] X a, 677
    DssVestSuckable.file(what, data) [DssVest] X a, 687
    DssVestTransferrable.file(what, data) [DssVest] X a, 697
  definition
    DssVestMintable.cap [DssVest] : uint256, 675
    DssVestSuckable.cap [DssVest] : uint256, 685
    DssVestTransferrable.cap [DssVest] : uint256, 695
  read
    DssVestMintable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a, 678
    DssVestSuckable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a, 688
    DssVestTransferrable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a, 698
cash
  call
    DssProxyActionsEnd.cashETH(ethJoin, end, ilk, wad) X, 544
    DssProxyActionsEnd.cashGem(gemJoin, end, ilk, wad) X, 545
  definition
    End.cash(ilk, wad) X, 301
cashETH
  definition
    DssProxyActionsEnd.cashETH(ethJoin, end, ilk, wad) X, 544
cashGem
  definition
    DssProxyActionsEnd.cashGem(gemJoin, end, ilk, wad) X, 545
cast
  definition
    DssExec.cast() X, 310

```

cat

- call
 - DssExecLib.setMaxTotalDAILiquidationAmountLEGACY(_amount) [X](#), 740
- write
 - End.file(what, [data](#)) [X](#) [a](#), 298
 - IlkRegistry.constructor(vat_, dog_, cat_, spot_) [X](#), 668
 - IlkRegistry.file(what, [data](#)) [X](#) [a](#), 669
- definition
 - DssExecLib.cat(), 730
 - End.cat: CatLike, 295
 - IlkRegistry.cat: CatLike, 667
- read
 - End.cage() [X](#) [a](#), 299
 - End.skip(ilk, id) [X](#), 299
 - IlkRegistry.add(adapter) [X](#), 668
 - IlkRegistry.constructor(vat_, dog_, cat_, spot_) [X](#), 668

cDai

- call
 - D3MInit.initCompound(dss, d3m, cfg, compoundCfg), 727
 - D3MInitScript.run() [X](#), 420
- write
 - D3MCompoundPlan.constructor(cDai_) [X](#), 315
 - D3MCompoundPool.constructor(ilk_, hub_, cDai_) [X](#), 377
- definition
 - D3MCompoundPlan.cDai: CErc20Like, 315
 - D3MCompoundPool.cDai: CErc20Like, 377
- read
 - D3MCompoundPlan.active(), 317
 - D3MCompoundPlan.constructor(cDai_) [X](#), 315
 - D3MCompoundPlan._calculateTargetSupply(targetInterestRate, borrows), 316
 - D3MCompoundPlan.getTargetAssets(currentAssets), 317
 - D3MCompoundPool.assetBalance(), 379
 - D3MCompoundPool.constructor(ilk_, hub_, cDai_) [X](#), 377
 - D3MCompoundPool.collect(claim) [X](#), 380
 - D3MCompoundPool.deposit(wad) [X](#), 378
 - D3MCompoundPool.exit(dst, wad) [X](#), 379
 - D3MCompoundPool.maxWithdraw(), 379
 - D3MCompoundPool.preDebtChange() [X](#), 379
 - D3MCompoundPool.quit(dst) [X](#) [a](#), 379
 - D3MCompoundPool.redeemable(), 380
 - D3MCompoundPool.withdraw(wad) [X](#), 379

cdpAllow

- call
 - DssProxyActions.cdpAllow(manager, cdp, usr, ok) [X](#), 532
- definition
 - DssCdpManager.cdpAllow(cdp, usr, ok) [X](#), 522
 - DssProxyActions.cdpAllow(manager, cdp, usr, ok) [X](#), 532

cdpAllowed

- call
 - DssCdpManager.cdpAllow(cdp, usr, ok) [X](#), 522
 - DssCdpManager.enter(src, cdp) [X](#), 525
 - DssCdpManager.flux(cdp, dst, wad) [X](#), 524
 - DssCdpManager.flux(ilk, cdp, dst, wad) [X](#), 524
 - DssCdpManager.frob(cdp, dink, dart) [X](#), 524
 - DssCdpManager.give(cdp, dst) [X](#), 523
 - DssCdpManager.move(cdp, dst, rad) [X](#), 524
 - DssCdpManager.quit(cdp, dst) [X](#), 524
 - DssCdpManager.shift(cdpSrc, cdpDst) [X](#), 525
- definition
 - DssCdpManager.modifier cdpAllowed(cdp), 521

cdpCan

- call
 - DssProxyActions.wipe(manager, daiJoin, cdp, wad) [X](#), 537

```

    DssProxyActions.wipeAll(manager, daiJoin, cdp) X, 537
write
    DssCdpManager.cdpAllow(cdp, usr, ok) X, 522
definition
    DssCdpManager.cdpCan: mapping(address => mapping(uint256 => mapping(address =>
        uint256))), 521
read
    DssCdpManager.modifier cdpAllowed(cdp), 521
cdpi
write
    DssCdpManager.open(ilk, usr) X, 522
definition
    DssCdpManager.cdpi: uint256, 521
read
    DssCdpManager.open(ilk, usr) X, 522
ceiling
write
    L1DAIBridge.setCeiling(_ceiling) X a, 619
definition
    L1DAIBridge.ceiling: uint256, 617
read
    L1DAIBridge.deposit(amount, l2Recipient) X, 619
cfg
write
    D3MInitScript.run() X, 420
definition
    D3MInitScript.cfg: D3MCommonConfig, 403
read
    D3MInitScript.run() X, 420
Chain
definition
    struct StdChains.Chain, 331, 359, 385, 405
ChainData
definition
    struct StdChains.ChainData, 331, 359, 385, 405
chainId
read
    D3MCoreDeployScript.setChain(chainAlias, chain) [StdChains], 344
    D3MCoreInitScript.setChain(chainAlias, chain) [StdChains], 372
    D3MDeployScript.setChain(chainAlias, chain) [StdChains], 398
    D3MInitScript.setChain(chainAlias, chain) [StdChains], 418
ChainLog
definition
    contract ChainLog, 709
chainlog
write
    DssVestSuckable.constructor(_chainlog) X, 692
definition
    DssVestSuckable.chainlog: ChainlogLike, 685
read
    DssVestSuckable.constructor(_chainlog) X, 692
    DssVestSuckable.pay(_guy, _amt), 693
chains
write
    D3MCoreDeployScript.setChain(chainAlias, chain) [StdChains], 344
    D3MCoreInitScript.setChain(chainAlias, chain) [StdChains], 372
    D3MDeployScript.setChain(chainAlias, chain) [StdChains], 398
    D3MInitScript.setChain(chainAlias, chain) [StdChains], 418
definition
    D3MCoreDeployScript.chains [StdChains] : mapping(string => StdChains.Chain), 330
    D3MCoreInitScript.chains [StdChains] : mapping(string => StdChains.Chain), 358
    D3MDeployScript.chains [StdChains] : mapping(string => StdChains.Chain), 384
    D3MInitScript.chains [StdChains] : mapping(string => StdChains.Chain), 404

```

read

D3MCoreDeployScript.getChain(chainAlias) [StdChains], 343
 D3MCoreDeployScript.getChain(chainId) [StdChains], 343
 D3MCoreDeployScript.setChain(chainAlias, chain) [StdChains], 344
 D3MCoreInitScript.getChain(chainAlias) [StdChains], 371
 D3MCoreInitScript.getChain(chainId) [StdChains], 371
 D3MCoreInitScript.setChain(chainAlias, chain) [StdChains], 372
 D3MDeployScript.getChain(chainAlias) [StdChains], 397
 D3MDeployScript.getChain(chainId) [StdChains], 397
 D3MDeployScript.setChain(chainAlias, chain) [StdChains], 398
 D3MInitScript.getChain(chainAlias) [StdChains], 417
 D3MInitScript.getChain(chainId) [StdChains], 418
 D3MInitScript.setChain(chainAlias, chain) [StdChains], 418

change

definition

OSM.change(src_) X a, 287

changeFees

definition

USDT.changeFees(_basisPointsRate, _maximumFee) X, 493

chi

write

Pot.constructor(vat_) X, 277

Pot.drip() X, 279

definition

Pot.chi: uint256, 277

read

Pot.drip() X, 279

Pot.exit(wad) X, 279

Pot.join(wad) X, 279

chip

write

Clipper.file(what, data) X a, 243

definition

Clipper.chip: uint64, 241

read

Clipper.kick(tab, lot, usr, kpr) X a, 244

Clipper.redo(id, kpr) X, 245

chop

call

Clipper.upchost() X, 248

write

Dog.file(ilk, what, data) X a, 253

definition

Dog.chop(ilk), 253

read

Dog.chop(ilk), 253

chost

write

Clipper.upchost() X, 248

definition

Clipper.chost: uint256, 241

read

Clipper.redo(id, kpr) X, 245

Clipper.take(id, amt, max, who, data) X, 246

claimComp

call

D3MCompoundPool.collect(claim) X, 380

claimRewards

call

D3MAavePool.collect() X, 352

class

write

IlkRegistry.file(ilk, what, data) X a, 670

```

definition
    IlkRegistry.class(ilk), 671
read
    IlkRegistry.class(ilk), 671
    IlkRegistry.put(_ilk, _join, _gem, _dec, _class, _pip, _xlip, _name, _symbol) X a,
        673
    IlkRegistry.remove(ilk) X, 669
    IlkRegistry.update(ilk) X, 673
clf
    write
        DssVestMintable._yank(_id, _end) [DssVest], 682
        DssVestSuckable._yank(_id, _end) [DssVest], 691
        DssVestTransferrable._yank(_id, _end) [DssVest], 701
    definition
        DssVestMintable.clf(_id) [DssVest], 676
        DssVestSuckable.clf(_id) [DssVest], 686
        DssVestTransferrable.clf(_id) [DssVest], 696
    read
        DssVestMintable.clf(_id) [DssVest], 676
        DssVestSuckable.clf(_id) [DssVest], 686
        DssVestTransferrable.clf(_id) [DssVest], 696
clip
    call
        DssExecLib.calc(_ilk), 732
        DssExecLib.setAuctionPermittedDrop(_ilk, _pct_bps) X, 745
        DssExecLib.setAuctionTimeBeforeReset(_ilk, _duration) X, 745
        DssExecLib.setIlkLiquidationPenalty(_ilk, _pct_bps) X, 743
        DssExecLib.setIlkMinVaultAmount(_ilk, _amount) X, 743
        DssExecLib.setKeeperIncentiveFlatRate(_ilk, _amount) X, 745
        DssExecLib.setKeeperIncentivePercent(_ilk, _pct_bps) X, 745
        DssExecLib.setStartingPriceMultiplicativeFactor(_ilk, _pct_bps) X, 744
    write
        Dog.file(ilk, what, clip) X a, 253
    definition
        DssExecLib.clip(_ilk), 732
Clipper
    definition
        contract Clipper, 241
clipperCall
    call
        Clipper.take(id, amt, max, who, data) X, 246
    definition
        CalleeMakerOtcDai.clipperCall(sender, daiAmt, gemAmt, data) X, 550
        CurveLpTokenUniv3Callee.clipperCall(sender, owe, slice, data) X, 552
        PSMCallee.clipperCall(sender, owe, slice, data) X, 555
        UniswapV2CalleeDai.clipperCall(sender, daiAmt, gemAmt, data) X, 558
        UniswapV2LpTokenCalleeDai.clipperCall(sender, daiAmt, gemAmt, data) X, 560
        UniswapV3Callee.clipperCall(sender, owe, slice, data) X, 567
        WstETHCurveUniv3Callee.clipperCall(sender, owe, slice, data) X, 564
ClipperMom
    definition
        contract ClipperMom, 257
clipperMom
    call
        DssExecLib.addNewCollateral(co) X, 750
        DssExecLib.setLiquidationBreakerPriceTolerance(_clip, _pct_bps) X, 745
    definition
        DssExecLib.clipperMom(), 732
close
    definition
        L1DAIBridge.close() X a, 618
        L1DaiGateway.close() X a, 571
        L1DAITokenBridge.close() X a, 594

```

L2DaiGateway.close() [X](#) [a](#), 582
 L2DAITokenBridge.close() [X](#) [a](#), 605

coins

 call

 CurveLpTokenUniv3Callee.clipperCall([sender](#), owe, slice, [data](#)) [X](#), 552

collect

 definition

 D3MAavePool.collect() [X](#), 352
 D3MCompoundPool.collect([claim](#)) [X](#), 380

COMP

 definition

 contract COMP, 457

comp

 call

 D3MInit.initCompound(dss, d3m, cfg, compoundCfg), 727
 D3MInitScript.run() [X](#), 420

 write

 D3MCompoundPool.constructor([ilk_](#), [hub_](#), [cDai_](#)) [X](#), 377

 definition

 D3MCompoundPool.comp: TokenLike, 377

 read

 D3MCompoundPool.constructor([ilk_](#), [hub_](#), [cDai_](#)) [X](#), 377
 D3MCompoundPool.collect([claim](#)) [X](#), 380

compoundCfg

 write

 D3MInitScript.run() [X](#), 420

 definition

 D3MInitScript.compoundCfg: D3MCompoundConfig, 403

 read

 D3MInitScript.run() [X](#), 420

comptroller

 call

 D3MCompoundPool.constructor([ilk_](#), [hub_](#), [cDai_](#)) [X](#), 377
 D3MInit.initCompound(dss, d3m, cfg, compoundCfg), 727
 D3MInitScript.run() [X](#), 420

 write

 D3MCompoundPool.constructor([ilk_](#), [hub_](#), [cDai_](#)) [X](#), 377

 definition

 D3MCompoundPool.comptroller: ComptrollerLike, 377

 read

 D3MCompoundPool.constructor([ilk_](#), [hub_](#), [cDai_](#)) [X](#), 377
 D3MCompoundPool.collect([claim](#)) [X](#), 380

computeCreate2Address

 definition

 D3MCoreDeployScript.computeCreate2Address(salt, initcodeHash, deployer) [StdUtils], 337
 D3MCoreInitScript.computeCreate2Address(salt, initcodeHash, deployer) [StdUtils], 365
 D3MDeployScript.computeCreate2Address(salt, initcodeHash, deployer) [StdUtils], 391
 D3MInitScript.computeCreate2Address(salt, initcodeHash, deployer) [StdUtils], 411

computeCreateAddress

 definition

 D3MCoreDeployScript.computeCreateAddress(deployer, nonce) [StdUtils], 336
 D3MCoreInitScript.computeCreateAddress(deployer, nonce) [StdUtils], 364
 D3MDeployScript.computeCreateAddress(deployer, nonce) [StdUtils], 390
 D3MInitScript.computeCreateAddress(deployer, nonce) [StdUtils], 410

config

 write

 D3MCoreDeployScript.run() [X](#), 346
 D3MCoreInitScript.run() [X](#), 374
 D3MDeployScript.run() [X](#), 400
 D3MInitScript.run() [X](#), 420

definition

D3MCoreDeployScript.config: [string](#), 329
 D3MCoreInitScript.config: [string](#), 357
 D3MDeployScript.config: [string](#), 383
 D3MInitScript.config: [string](#), 403

read

D3MCoreDeployScript.run() [X](#), 346
 D3MCoreInitScript.run() [X](#), 374
 D3MDeployScript.run() [X](#), 400
 D3MInitScript.run() [X](#), 420

CONSOLE

definition

D3MCoreDeployScript.CONSOLE [CommonBase] : [address](#), 329
 D3MCoreInitScript.CONSOLE [CommonBase] : [address](#), 357
 D3MDeployScript.CONSOLE [CommonBase] : [address](#), 383
 D3MInitScript.CONSOLE [CommonBase] : [address](#), 403

CONSOLE2_ADDRESS

definition

D3MCoreDeployScript.CONSOLE2_ADDRESS [StdUtils] : [address](#), 329
 D3MCoreInitScript.CONSOLE2_ADDRESS [StdUtils] : [address](#), 357
 D3MDeployScript.CONSOLE2_ADDRESS [StdUtils] : [address](#), 383
 D3MInitScript.CONSOLE2_ADDRESS [StdUtils] : [address](#), 403

read

D3MCoreDeployScript.console2_log(p0, p1) [StdUtils], 337, 338
 D3MCoreInitScript.console2_log(p0, p1) [StdUtils], 365, 366
 D3MDeployScript.console2_log(p0, p1) [StdUtils], 391, 392
 D3MInitScript.console2_log(p0, p1) [StdUtils], 412

console2_log

call

D3MCoreDeployScript.bound(x, min, max) [StdUtils], 336
 D3MCoreInitScript.bound(x, min, max) [StdUtils], 364
 D3MDeployScript.bound(x, min, max) [StdUtils], 390
 D3MInitScript.bound(x, min, max) [StdUtils], 410

definition

D3MCoreDeployScript.console2_log(p0, p1) [StdUtils], 337, 338
 D3MCoreInitScript.console2_log(p0, p1) [StdUtils], 365, 366
 D3MDeployScript.console2_log(p0, p1) [StdUtils], 391, 392
 D3MInitScript.console2_log(p0, p1) [StdUtils], 412

constructor

call

DAIMock.constructor() [X](#), 632
 DssVestMintable.constructor(_gem) [X](#), 683
 DssVestSuckable.constructor(_chainlog) [X](#), 692
 DssVestTransferrable.constructor(_czar, _gem) [X](#), 703
 L1DaiGateway.constructor(_l2Counterpart, _l1Router, _inbox, _l1Dai, _l2Dai, _l1Escrow) [X](#), 571
 L1DAITokenBridge.constructor(_l1Token, _l2DAITokenBridge, _l2Token, _l1messenger, _escrow) [X](#), 593
 L1GovernanceRelay.constructor(_inbox, _l2GovernanceRelay) [X](#), 579
 L1GovernanceRelay.constructor(_l2GovernanceRelay, _l1messenger) [X](#), 601
 L2DAITokenBridge.constructor(_l2CrossDomainMessenger, _l2Token, _l1Token, _l1DAITokenBridge) [X](#), 605
 L2GovernanceRelay.constructor(_l2CrossDomainMessenger, _l1GovernanceRelay) [X](#), 610

definition

AAVE.constructor(supply) [X](#), 451
 AuthGemJoin5.constructor(vat_, ilk_, gem_) [X](#), 503
 AuthGemJoin8.constructor(vat_, ilk_, gem_) [X](#), 505
 AuthGemJoin.constructor(vat_, ilk_, gem_) [X](#), 447, 507
 BAL.constructor(supply) [X](#), 453
 BasicRelay.constructor(_oracleAuth, _daiJoin) [X](#), 657
 BAT.constructor(supply) [X](#), 455
 CalleeMakerOtcDai.constructor(otc_, daiJoin_) [X](#), 550
 ChainLog.constructor() [X](#), 709

Clipper.constructor(vat_, spotter_, dog_, ilk_) X, 243
 ClipperMom.constructor(spotter_) X, 257
 COMP.constructor(supply) X, 457
 Cure.constructor() X, 292
 CurveLpTokenUniv3Callee.constructor(uniV3Router_, daiJoin_, weth_) X, 551
 D3MAavePlan.constructor(dai_, pool_) X, 311
 D3MAavePool.constructor(ilk_, hub_, dai_, pool_) X, 349
 D3MCompoundPlan.constructor(cDai_) X, 315
 D3MCompoundPool.constructor(ilk_, hub_, cDai_) X, 377
 D3MHub.constructor(daiJoin_) X, 320
 D3MMom.constructor() X, 355
 D3MOracle.constructor(vat_, ilk_) X, 353
 Dai.constructor() X, 588, 614
 Dai.constructor(chainId_) X, 224
 DaiJoin.constructor(vat_, dai_) X, 425
 DAIMock.constructor() X, 632
 DAIMock.constructor(name_, symbol_) [ERC20] X, 626
 DGD.constructor(supply) X, 459
 Dog.constructor(vat_) X, 252
 DssAutoLine.constructor(vat_) X, 705
 DssCdpManager.constructor(vat_) X, 522
 DssExec.constructor(_expiration, _spellAction) X, 309
 DssPsm.constructor(gemJoin_, daiJoin_, vow_) X, 509
 DssVestMintable.constructor() [DssVest] X, 677
 DssVestMintable.constructor(_gem) X, 683
 DssVestSuckable.constructor() [DssVest] X, 687
 DssVestSuckable.constructor(_chainlog) X, 692
 DssVestTransferrable.constructor() [DssVest] X, 697
 DssVestTransferrable.constructor(_czar, _gem) X, 703
 End.constructor() X, 297
 ESM.constructor(gem_, end_, proxy_, min_) X, 303
 ExponentialDecrease.constructor() X, 239
 Flapper.constructor(vat_, gem_) X, 262
 Flopper.constructor(vat_, gem_) X, 266
 GemBag.constructor(lad_, gem_) X, 431
 GemJoin2.constructor(vat_, ilk_, gem_) X, 427
 GemJoin3.constructor(vat_, ilk_, gem_, decimals) X, 429
 GemJoin4.constructor(vat_, ilk_, gem_) X, 433
 GemJoin5.constructor(vat_, ilk_, gem_) X, 435
 GemJoin6.constructor(vat_, ilk_, gem_) X, 437
 GemJoin7.constructor(vat_, ilk_, gem_) X, 439
 GemJoin8.constructor(vat_, ilk_, gem_) X, 441
 GemJoin9.constructor(vat_, ilk_, gem_) X, 443
 GemJoin.constructor(vat_, ilk_, gem_) X, 424
 GNT.constructor(supply) X, 461
 GUSD.constructor(supply) X, 463
 IlkRegistry.constructor(vat_, dog_, cat_, spot_) X, 668
 Jug.constructor(vat_) X, 273
 KNC.constructor(supply) X, 465
 L1DAIBridge.constructor(_starkNet, _dai, _l2Dai, _escrow, _l2DaiBridge) X, 618
 L1DaiGateway.constructor(_inbox) [L1CrossDomainEnabled] X, 570
 L1DaiGateway.constructor(_l2Counterpart, _l1Router, _inbox, _l1Dai, _l2Dai, _l1Escrow) X, 571
 L1DAITeleportGateway.constructor(_starkNet, _l1Token, _l2TeleportGateway, _l1Escrow, _l1TeleportRouter) X, 623
 L1DAITokenBridge.constructor(_l1Token, _l2DAITokenBridge, _l2Token, _l1messenger, _escrow) X, 593
 L1DAITokenBridge.constructor(_messenger) [OVM.CrossDomainEnabled] X, 592
 L1Escrow.constructor() X, 575, 597, 635
 L1EscrowMom.constructor(escrow_, token_) X, 637
 L1GovernanceRelay.constructor(_inbox) [L1CrossDomainEnabled] X, 577
 L1GovernanceRelay.constructor(_inbox, _l2GovernanceRelay) X, 579
 L1GovernanceRelay.constructor(_l2GovernanceRelay, _l1messenger) X, 601

L1GovernanceRelay.constructor(_messenger) [OVM_CrossDomainEnabled] X, 600
 L1GovernanceRelay.constructor(_starkNet, _l2GovernanceRelay) X, 621
 L2DaiGateway.constructor(_l1Counterpart, _l2Router, _l1Dai, _l2Dai) X, 582
 L2DAITokenBridge.constructor(_l2CrossDomainMessenger, _l2Token, _l1Token, _l1DAITokenBridge) X, 605
 L2DAITokenBridge.constructor(_messenger) [OVM_CrossDomainEnabled] X, 604
 L2GovernanceRelay.constructor(_l1GovernanceRelay) X, 585
 L2GovernanceRelay.constructor(_l2CrossDomainMessenger, _l1GovernanceRelay) X, 610
 L2GovernanceRelay.constructor(_messenger) [OVM_CrossDomainEnabled] X, 610
 LinearDecrease.constructor() X, 235
 LINK.constructor(supply) X, 467
 LRC.constructor(supply) X, 469
 MANA.constructor(supply) X, 471
 ManagedGemJoin.constructor(vat_, ilk_, gem_) X, 449
 MATIC.constructor(supply) X, 473
 Median.constructor() X, 282
 OMG.constructor(supply) X, 475
 OSM.constructor(src_) X, 286
 PAXG.constructor(supply, fee) X, 478
 PAXUSD.constructor(supply) X, 481
 Pot.constructor(vat_) X, 277
 PSMCallee.constructor(daiJoin_) X, 555
 RENBTC.constructor(supply) X, 483
 REP.constructor(supply) X, 485
 Spotter.constructor(vat_) X, 227
 StairstepExponentialDecrease.constructor() X, 237
 TeleportConstantFee.constructor(_fee, _ttl) X, 639
 TeleportJoin.constructor(vat_, daiJoin_, ilk_, domain_) X, 641
 TeleportLinearFee.constructor(_fee, _ttl) X, 647
 TeleportOracleAuth.constructor(teleportJoin_) X, 649
 TeleportRouter.constructor(dai_) X, 653
 TrustedRelay.constructor(_oracleAuth, _daiJoin, _ethPriceOracle) X, 661
 TUSD.constructor(supply) X, 487
 UNI.constructor(supply) X, 489
 UniswapV2CalleeDai.constructor(uniRouter02_, daiJoin_) X, 557
 UniswapV2LpTokenCalleeDai.constructor(uniRouter02_, daiJoin_) X, 559
 UniswapV3Callee.constructor(uniV3Router_, daiJoin_) X, 567
 UrnHandler.constructor(vat) X, 519
 USDC.constructor(supply) X, 491
 USDT.constructor(_initialSupply) X, 493
 Vat.constructor() X, 230
 Vow.constructor(vat_, flapper_, flopper_) X, 269
 WBTC.constructor(supply) X, 495
 WSTETH.constructor(supply) X, 497
 WstETHCurveUniv3Callee.constructor(curvePool_, uniV3Router_, daiJoin_, weth_) X, 563
 YFI.constructor(supply) X, 499
 ZRX.constructor(supply) X, 501
 consumeMessageFromL2
 call
 L1DAIBridge.withdraw(amount, l1Recipient) X, 619
 L1DAITeleportGateway.finalizeFlush(targetDomain, daiToFlush) X, 623
 L1DAITeleportGateway.finalizeRegisterTeleport(teleport) X, 623
 _contains
 call
 EnumerableSet._add(set, value), 756
 EnumerableSet.contains(set, value), 758, 759, 761
 TeleportRouter.hasDomain(domain), 654
 definition
 EnumerableSet._contains(set, value), 757–759, 761
 convertTo18
 call
 DssProxyActions.exitGem(manager, gemJoin, cdp, amt) X, 536

DssProxyActions.freeGem(manager, gemJoin, cdp, amt) [X](#), 535
 DssProxyActions.lockGem(manager, gemJoin, cdp, amt, transferFrom) [X](#), 535
 DssProxyActions.lockGemAndDraw(manager, jug, gemJoin, daiJoin, cdp, amtC, wadD, transferFrom) [X](#), 539
 DssProxyActions.wipeAllAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC) [X](#), 541
 DssProxyActions.wipeAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC, wadD) [X](#), 541
 definition
 DssProxyActions.convertTo18(gemJoin, amt), 530
 count
 call
 ChainLog.setAddress(_key, _addr) [X](#) [a](#), 710
 write
 DssCdpManager.give(cdp, dst) [X](#), 523
 DssCdpManager.open(ilk, usr) [X](#), 522
 definition
 ChainLog.count(), 711
 Clipper.count(), 248
 DssCdpManager.count: [mapping](#)(address => uint256), 521
 IlkRegistry.count(), 670
 read
 DssCdpManager.give(cdp, dst) [X](#), 523
 DssCdpManager.open(ilk, usr) [X](#), 522
 GetCdps.getCdpsAsc(manager, guy), 527
 GetCdps.getCdpsDesc(manager, guy), 527
 counterpartGateway
 definition
 L1DaiGateway.counterpartGateway(), 573
 L2DaiGateway.counterpartGateway(), 584
 create
 definition
 DssVestMintable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] [X](#) [a](#), 678
 DssVestSuckable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] [X](#) [a](#), 688
 DssVestTransferrable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] [X](#) [a](#), 698
 CREATE2_FACTORY
 definition
 D3MCoreDeployScript.CREATE2_FACTORY [ScriptBase] : [address](#), 329
 D3MCoreInitScript.CREATE2_FACTORY [ScriptBase] : [address](#), 357
 D3MDeployScript.CREATE2_FACTORY [ScriptBase] : [address](#), 383
 D3MInitScript.CREATE2_FACTORY [ScriptBase] : [address](#), 403
 createRetryableTicket
 call
 L1DaiGateway.sendTxToL2(target, user, maxSubmissionCost, maxGas, gasPriceBid, [data](#)) [L1CrossDomainEnabled], 570
 L1GovernanceRelay.sendTxToL2(target, user, maxSubmissionCost, maxGas, gasPriceBid, [data](#)) [L1CrossDomainEnabled], 577
 createRetryableTicketNoRefundAliasRewrite
 call
 L1DaiGateway.sendTxToL2NoAliasing(target, user, l1CallValue, maxSubmissionCost, maxGas, gasPriceBid, [data](#)) [L1CrossDomainEnabled], 570
 L1GovernanceRelay.sendTxToL2NoAliasing(target, user, l1CallValue, maxSubmissionCost, maxGas, gasPriceBid, [data](#)) [L1CrossDomainEnabled], 578
 cull
 definition
 D3MHub.cull(ilk) [X](#), 326
 culled
 call
 D3MOracle.peek(), 354
 write
 D3MHub.cull(ilk) [X](#), 326
 D3MHub.uncull(ilk) [X](#), 326
 definition
 D3MHub.culled(ilk), 327
 read

```

    D3MHub.cull(ilk) X, 326
    D3MHub.culled(ilk), 327
    D3MHub.exec(ilk) X, 324
    D3MHub.uncull(ilk) X, 326
cur
  write
    OSM.poke() X, 287
    OSM.void() X a, 287
  definition
    OSM.cur: OSM.Feed, 285
  read
    OSM.peek(), 287
    OSM.poke() X, 287
    OSM.read(), 288
Cure
  definition
    contract Cure, 291
cure
  call
    Cure.load(src) X, 293
  write
    End.file(what, data) X a, 298
  definition
    End.cure: CureLike, 295
    TeleportJoin.cure(), 643
  read
    End.cage() X a, 299
    End.thaw() X, 300
CurveData
  definition
    struct CurveLpTokenUniv3Callee.CurveData, 551
CurveLpTokenUniv3Callee
  definition
    contract CurveLpTokenUniv3Callee, 551
curvePool
  write
    WstETHCurveUniv3Callee.constructor(curvePool_, univ3Router_, daiJoin_, weth_) X,
    563
  definition
    WstETHCurveUniv3Callee.curvePool: CurvePoolLike, 563
  read
    WstETHCurveUniv3Callee.clipperCall(sender, owe, slice, data) X, 564
cusp
  write
    Clipper.file(what, data) X a, 243
  definition
    Clipper.cusp: uint256, 241
  read
    Clipper.status(tic, top), 248
cut
  write
    ExponentialDecrease.file(what, data) X a, 239
    StairstepExponentialDecrease.file(what, data) X a, 237
  definition
    ExponentialDecrease.cut: uint256, 239
    StairstepExponentialDecrease.cut: uint256, 237
  read
    ExponentialDecrease.file(what, data) X a, 239
    ExponentialDecrease.price(top, dur), 240
    StairstepExponentialDecrease.file(what, data) X a, 237
    StairstepExponentialDecrease.price(top, dur), 238
czar
  write

```

DssVestTransferrable.[constructor](#)(_czar, _gem) [X](#), 703

[definition](#)

DssVestTransferrable.czar: [address](#), 695

[read](#)

DssVestTransferrable.pay(_guy, _amt), 703

D**d3m****write**

D3MDeployScript.run() **X**, 400

D3MInitScript.run() **X**, 420

definition

D3MDeployScript.d3m: D3MInstance, 383

D3MInitScript.d3m: D3MInstance, 403

read

D3MDeployScript.run() **X**, 400

D3MInitScript.run() **X**, 420

D3MAavePlan**definition**

contract D3MAavePlan, 311

D3MAavePool**definition**

contract D3MAavePool, 349

D3MCompoundPlan**definition**

contract D3MCompoundPlan, 315

D3MCompoundPool**definition**

contract D3MCompoundPool, 377

d3mCore**write**

D3MCoreDeployScript.run() **X**, 346

D3MCoreInitScript.run() **X**, 374

definition

D3MCoreDeployScript.d3mCore: D3MCoreInstance, 329

D3MCoreInitScript.d3mCore: D3MCoreInstance, 357

read

D3MCoreDeployScript.run() **X**, 346

D3MCoreInitScript.run() **X**, 374

D3MCoreDeployScript**definition**

contract D3MCoreDeployScript, 329

D3MCoreInitScript**definition**

contract D3MCoreInitScript, 357

D3MDeploy**definition**

library D3MDeploy, 723

D3MDeployScript**definition**

contract D3MDeployScript, 383

D3MHub**definition**

contract D3MHub, 319

D3MInit**definition**

library D3MInit, 725

D3MInitScript**definition**

contract D3MInitScript, 403

D3MMom**definition**

contract D3MMom, 355

D3MOracle**definition**

contract D3MOracle, 353

d3mType**write**

D3MDeployScript.run() **X**, 400

D3MInitScript.run() [X](#), 420

definition

D3MDeployScript.d3mType: [string](#), 383

D3MInitScript.d3mType: [string](#), 403

read

D3MDeployScript.run() [X](#), 400

D3MInitScript.run() [X](#), 420

Dai

definition

[contract](#) Dai, 223

[contract](#) Dai(2), 587

[contract](#) Dai(3), 613

dai

call

BasicRelay.constructor(_oracleAuth, _daiJoin) [X](#), 657

CalleeMakerOtcDai.setUp(otc_, daiJoin_) [CalleeMakerOtc], 549

CurveLpTokenUniv3Callee.constructor(uniV3Router_, daiJoin_, weth_) [X](#), 551

D3MHub.constructor(daiJoin_) [X](#), 320

D3MInit.initAave(dss, d3m, cfg, aaveCfg), 726

D3MInit.initCompound(dss, d3m, cfg, compoundCfg), 727

DssProxyActions.daiJoin_join(apt, urn, wad) [Common] [X](#), 529

DssProxyActionsDsr.daiJoin_join(apt, urn, wad) [Common] [X](#), 547

DssProxyActionsDsr.exit(daiJoin, pot, wad) [X](#), 548

DssProxyActionsEnd.daiJoin_join(apt, urn, wad) [Common] [X](#), 543

DssProxyActions._getDrawDart(vat, jug, urn, ilk, wad), 530

DssProxyActions._getWipeAllWad(vat, usr, urn, ilk), 530

DssProxyActions.wipe(manager, daiJoin, cdp, wad) [X](#), 537

DssProxyActions.wipeAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC, wadD) [X](#), 540

DssProxyActions.wipeAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC, wadD) [X](#), 541

DssPsm.constructor(gemJoin_, daiJoin_, vow_) [X](#), 509

End.thaw() [X](#), 300

PSMCallee.constructor(daiJoin_) [X](#), 555

TeleportJoin.constructor(vat_, daiJoin_, ilk_, domain_) [X](#), 641

TrustedRelay.constructor(_oracleAuth, _daiJoin, _ethPriceOracle) [X](#), 661

UniswapV2CalleeDai.setUp(uniRouter02_, daiJoin_) [UniswapV2Callee], 557

UniswapV2LpTokenCalleeDai.setUp(uniRouter02_, daiJoin_) [UniswapV2Callee(2)], 559

UniswapV3Callee.constructor(uniV3Router_, daiJoin_) [X](#), 567

Vow.cage() [X](#) [a](#), 271

Vow.flap() [X](#), 271

Vow.flop() [X](#), 271

Vow.heal(rad) [X](#), 270

Vow.kiss(rad) [X](#), 271

WstETHCurveUniv3Callee.constructor(curvePool_, uniV3Router_, daiJoin_, weth_) [X](#), 563

write

BasicRelay.constructor(_oracleAuth, _daiJoin) [X](#), 657

CalleeMakerOtcDai.setUp(otc_, daiJoin_) [CalleeMakerOtc], 549

CurveLpTokenUniv3Callee.constructor(uniV3Router_, daiJoin_, weth_) [X](#), 551

D3MAavePlan.constructor(dai_, pool_) [X](#), 311

D3MAavePool.constructor(ilk_, hub_, dai_, pool_) [X](#), 349

D3MCompoundPool.constructor(ilk_, hub_, cDai_) [X](#), 377

DaiJoin.constructor(vat_, dai_) [X](#), 425

DssPsm.constructor(gemJoin_, daiJoin_, vow_) [X](#), 509

L1DAIBridge.constructor(_starkNet, _dai, _l2Dai, _escrow, _l2DaiBridge) [X](#), 618

PSMCallee.constructor(daiJoin_) [X](#), 555

TeleportRouter.constructor(dai_) [X](#), 653

TrustedRelay.constructor(_oracleAuth, _daiJoin, _ethPriceOracle) [X](#), 661

UniswapV2CalleeDai.setUp(uniRouter02_, daiJoin_) [UniswapV2Callee], 557

UniswapV2LpTokenCalleeDai.setUp(uniRouter02_, daiJoin_) [UniswapV2Callee(2)], 559

UniswapV3Callee.constructor(uniV3Router_, daiJoin_) [X](#), 567

Vat.fold(i, u, rate) [X](#) [a](#), 234

Vat.frob(i, u, v, w, dink, dart) [X](#), 232

Vat.heal(rad) [X](#), 233

Vat.move(src, dst, rad) [X](#), 232
 Vat.suck(u, v, rad) [X](#) [a](#), 234
 WstETHCurveUniv3Callee.constructor(curvePool_, uniV3Router_, daiJoin_, weth_) [X](#), 563

definition

BasicRelay.dai: TokenLike, 657
 CalleeMakerOtcDai.dai [CalleeMakerOtc] : TokenLike, 549
 CurveLpTokenUniv3Callee.dai: TokenLike, 551
 D3MAavePlan.dai: TokenLike, 311
 D3MAavePool.dai: TokenLike, 349
 D3MCompoundPool.dai: TokenLike, 377
 DaiJoin.dai: DSTokenLike, 425
 DssExecLib.dai(), 730
 DssPsm.dai: DaiAbstract, 509
 L1DAIBridge.dai: [address](#), 617
 PSMCallee.dai: TokenLike, 555
 TeleportRouter.dai: TokenLike, 653
 TrustedRelay.dai: TokenLike, 661
 UniswapV2CalleeDai.dai [UniswapV2Callee] : TokenLike, 557
 UniswapV2LpTokenCalleeDai.dai [UniswapV2Callee(2)] : TokenLike, 559
 UniswapV3Callee.dai: TokenLike, 567
 Vat.dai: [mapping](#)([address](#) => [uint256](#)), 229
 WstETHCurveUniv3Callee.dai: TokenLike, 563

read

BasicRelay.relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s) [X](#), 658
 CalleeMakerOtcDai.clipperCall(sender, daiAmt, gemAmt, data) [X](#), 550
 CalleeMakerOtcDai.setUp(otc_, daiJoin_) [CalleeMakerOtc], 549
 CurveLpTokenUniv3Callee.clipperCall(sender, owe, slice, data) [X](#), 552
 D3MAavePlan.active(), 314
 D3MAavePlan.getTargetAssets(currentAssets), 313
 D3MAavePool.constructor(ilk_, hub_, dai_, pool_) [X](#), 349
 D3MAavePool.deposit(wad) [X](#), 350
 D3MAavePool.maxWithdraw(), 351
 D3MAavePool.withdraw(wad) [X](#), 351
 D3MCompoundPool.constructor(ilk_, hub_, cDai_) [X](#), 377
 D3MCompoundPool.withdraw(wad) [X](#), 379
 DaiJoin.exit(usr, wad) [X](#), 426
 DaiJoin.join(usr, wad) [X](#), 426
 DssPsm.constructor(gemJoin_, daiJoin_, vow_) [X](#), 509
 DssPsm.buyGem(usr, gemAmt) [X](#), 511
 L1DAIBridge.cancelDeposit(amount, l2Recipient, l1Recipient, nonce) [X](#), 620
 L1DAIBridge.deposit(amount, l2Recipient) [X](#), 619
 L1DAIBridge.withdraw(amount, l1Recipient) [X](#), 619
 PSMCallee.constructor(daiJoin_) [X](#), 555
 PSMCallee.clipperCall(sender, owe, slice, data) [X](#), 555
 TeleportRouter.settle(targetDomain, batchedDaiToFlush) [X](#), 655
 TrustedRelay.relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s) [X](#), 663
 UniswapV2CalleeDai.clipperCall(sender, daiAmt, gemAmt, data) [X](#), 558
 UniswapV2CalleeDai.setUp(uniRouter02_, daiJoin_) [UniswapV2Callee], 557
 UniswapV2LpTokenCalleeDai.clipperCall(sender, daiAmt, gemAmt, data) [X](#), 560
 UniswapV2LpTokenCalleeDai.setUp(uniRouter02_, daiJoin_) [UniswapV2Callee(2)], 559
 UniswapV3Callee.constructor(uniV3Router_, daiJoin_) [X](#), 567
 UniswapV3Callee.clipperCall(sender, owe, slice, data) [X](#), 567
 Vat.fold(i, u, rate) [X](#) [a](#), 234
 Vat.frob(i, u, v, w, dink, dart) [X](#), 232
 Vat.heal(rad) [X](#), 233
 Vat.move(src, dst, rad) [X](#), 232
 Vat.suck(u, v, rad) [X](#) [a](#), 234
 WstETHCurveUniv3Callee.clipperCall(sender, owe, slice, data) [X](#), 564

DaiJoin

definition

`contract` DaiJoin, 425

daiJoin

call

- D3MInit.initCore(dss, d3mCore), 725
- DssExecLib.sendPaymentFromSurplusBuffer(_target, _amount) **X**, 751

write

- BasicRelay.constructor(_oracleAuth, _daiJoin) **X**, 657
- CalleeMakerOtcDai.setUp(otc_, daiJoin_) [CalleeMakerOtc], 549
- CurveLpTokenUniv3Callee.constructor(uniV3Router_, daiJoin_, weth_) **X**, 551
- D3MHub.constructor(daiJoin_) **X**, 320
- DssPsm.constructor(gemJoin_, daiJoin_, vow_) **X**, 509
- DssVestSuckable.constructor(_chainlog) **X**, 692
- PSMCallee.constructor(daiJoin_) **X**, 555
- TeleportJoin.constructor(vat_, daiJoin_, ilk_, domain_) **X**, 641
- TrustedRelay.constructor(_oracleAuth, _daiJoin, _ethPriceOracle) **X**, 661
- UniswapV2CalleeDai.setUp(uniRouter02_, daiJoin_) [UniswapV2Callee], 557
- UniswapV2LpTokenCalleeDai.setUp(uniRouter02_, daiJoin_) [UniswapV2Callee(2)], 559
- UniswapV3Callee.constructor(uniV3Router_, daiJoin_) **X**, 567
- WstETHCurveUniv3Callee.constructor(curvePool_, uniV3Router_, daiJoin_, weth_) **X**, 563

definition

- BasicRelay.daiJoin: DaiJoinLike, 657
- CalleeMakerOtcDai.daiJoin [CalleeMakerOtc] : DaiJoinLike, 549
- CurveLpTokenUniv3Callee.daiJoin: DaiJoinLike, 551
- D3MHub.daiJoin: DaiJoinLike, 319
- DssExecLib.daiJoin(), 732
- DssPsm.daiJoin: DaiJoinAbstract, 509
- DssVestSuckable.daiJoin: DaiJoinLike, 685
- PSMCallee.daiJoin: DaiJoinLike, 555
- TeleportJoin.daiJoin: DaiJoinLike, 641
- TrustedRelay.daiJoin: DaiJoinLike, 661
- UniswapV2CalleeDai.daiJoin [UniswapV2Callee] : DaiJoinLike, 557
- UniswapV2LpTokenCalleeDai.daiJoin [UniswapV2Callee(2)] : DaiJoinLike, 559
- UniswapV3Callee.daiJoin: DaiJoinLike, 567
- WstETHCurveUniv3Callee.daiJoin: DaiJoinLike, 563

read

- BasicRelay.constructor(_oracleAuth, _daiJoin) **X**, 657
- CalleeMakerOtcDai.clipperCall(sender, daiAmt, gemAmt, data) **X**, 550
- CalleeMakerOtcDai.setUp(otc_, daiJoin_) [CalleeMakerOtc], 549
- CurveLpTokenUniv3Callee.clipperCall(sender, owe, slice, data) **X**, 552
- D3MHub.constructor(daiJoin_) **X**, 320
- D3MHub._exec(ilk, _pool, Art, lineWad), 322
- D3MHub._wipe(ilk, _pool, urn), 322
- DssPsm.constructor(gemJoin_, daiJoin_, vow_) **X**, 509
- DssPsm.buyGem(usr, gemAmt) **X**, 511
- DssPsm.sellGem(usr, gemAmt) **X**, 510
- DssVestSuckable.constructor(_chainlog) **X**, 692
- DssVestSuckable.pay(_guy, _amt), 693
- PSMCallee.constructor(daiJoin_) **X**, 555
- PSMCallee.clipperCall(sender, owe, slice, data) **X**, 555
- TeleportJoin.constructor(vat_, daiJoin_, ilk_, domain_) **X**, 641
- TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643
- TeleportJoin.settle(sourceDomain, batchedDaiToFlush) **X**, 645
- TrustedRelay.constructor(_oracleAuth, _daiJoin, _ethPriceOracle) **X**, 661
- UniswapV2CalleeDai.clipperCall(sender, daiAmt, gemAmt, data) **X**, 558
- UniswapV2CalleeDai.setUp(uniRouter02_, daiJoin_) [UniswapV2Callee], 557
- UniswapV2LpTokenCalleeDai.clipperCall(sender, daiAmt, gemAmt, data) **X**, 560
- UniswapV2LpTokenCalleeDai.setUp(uniRouter02_, daiJoin_) [UniswapV2Callee(2)], 559
- UniswapV3Callee.constructor(uniV3Router_, daiJoin_) **X**, 567
- UniswapV3Callee.clipperCall(sender, owe, slice, data) **X**, 567
- WstETHCurveUniv3Callee.clipperCall(sender, owe, slice, data) **X**, 564

daiJoin_.join

call

DssProxyActionsDsr.join(daiJoin, pot, wad) [X](#), 547
 DssProxyActionsEnd.pack(daiJoin, end, wad) [X](#), 544
 DssProxyActions.wipe(manager, daiJoin, cdp, wad) [X](#), 537
 DssProxyActions.wipeAll(manager, daiJoin, cdp) [X](#), 537
 DssProxyActions.wipeAllAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC) [X](#), 540
 DssProxyActions.wipeAllAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC) [X](#), 541
 DssProxyActions.wipeAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC, wadD) [X](#), 540
 DssProxyActions.wipeAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC, wadD) [X](#), 541
 definition
 DssProxyActions.daiJoin_join(apt, urn, wad) [Common] [X](#), 529
 DssProxyActionsDsr.daiJoin_join(apt, urn, wad) [Common] [X](#), 547
 DssProxyActionsEnd.daiJoin_join(apt, urn, wad) [Common] [X](#), 543
 DAIMock
 definition
 contract DAIMock, 625
 data
 write
 D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341
 D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369
 D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395
 D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415
 read
 D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341
 D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369
 D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395
 D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415
 deal
 definition
 Flapper.deal(id) [X](#), 264
 Flopper.deal(id) [X](#), 267
 deauthorize
 definition
 DssExecLib.deauthorize(_base, _ward) [X](#), 734
 debt
 call
 D3MHub.exec(ilk) [X](#), 324
 D3MHub._exec(ilk, _pool, Art, lineWad), 322
 End.thaw() [X](#), 300
 write
 End.thaw() [X](#), 300
 TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643
 TeleportJoin.settle(sourceDomain, batchedDaiToFlush) [X](#), 645
 Vat.fold(i, u, rate) [X](#) [a](#), 234
 Vat.frob(i, u, v, w, dink, dart) [X](#), 232
 Vat.heal(rad) [X](#), 233
 Vat.suck(u, v, rad) [X](#) [a](#), 234
 definition
 End.debt: uint256, 295
 TeleportJoin.debt: mapping(bytes32 => int256), 641
 Vat.debt: uint256, 229
 read
 End.flow(ilk) [X](#), 300
 End.pack(wad) [X](#), 301
 End.thaw() [X](#), 300
 TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643
 TeleportJoin.settle(sourceDomain, batchedDaiToFlush) [X](#), 645
 Vat.fold(i, u, rate) [X](#) [a](#), 234
 Vat.frob(i, u, v, w, dink, dart) [X](#), 232
 Vat.heal(rad) [X](#), 233
 Vat.suck(u, v, rad) [X](#) [a](#), 234
 dec
 call
 CalleeMakerOtcDai._fromWad(gemJoin, wad) [CalleeMakerOtc], 550

CurveLpTokenUniv3Callee.fromWad(gemJoin, wad), 552
 DssExecLib.addCollateralBase(_ilk, _gem, _join, _clip, _calc, _pip) **X**, 749
 DssProxyActions.convertTo18(gemJoin, amt), 530
 DssProxyActionsEnd.cashGem(gemJoin, end, ilk, wad) **X**, 545
 DssProxyActionsEnd.freeGem(manager, gemJoin, end, cdp) **X**, 544
 DssPsm.constructor(gemJoin_, daiJoin_, vow_) **X**, 509
 IlkRegistry.add(adapter) **X**, 668
 PSMCallee.fromWad(gemJoin, wad), 555
 UniswapV2CalleeDai.fromWad(gemJoin, wad) [UniswapV2Callee], 557
 UniswapV2LpTokenCalleeDai.fromWad(gemJoin, wad) [UniswapV2Callee(2)], 559
 UniswapV3Callee.fromWad(gemJoin, wad), 567
 WstETHCurveUniv3Callee.fromWad(gemJoin, wad), 563

write

AuthGemJoin5.constructor(vat_, ilk_, gem_) **X**, 503
 AuthGemJoin8.constructor(vat_, ilk_, gem_) **X**, 505
 AuthGemJoin.constructor(vat_, ilk_, gem_) **X**, 447, 507
 GemJoin2.constructor(vat_, ilk_, gem_) **X**, 427
 GemJoin3.constructor(vat_, ilk_, gem_, decimals) **X**, 429
 GemJoin4.constructor(vat_, ilk_, gem_) **X**, 433
 GemJoin5.constructor(vat_, ilk_, gem_) **X**, 435
 GemJoin6.constructor(vat_, ilk_, gem_) **X**, 437
 GemJoin7.constructor(vat_, ilk_, gem_) **X**, 439
 GemJoin8.constructor(vat_, ilk_, gem_) **X**, 441
 GemJoin9.constructor(vat_, ilk_, gem_) **X**, 443
 GemJoin.constructor(vat_, ilk_, gem_) **X**, 424
 IlkRegistry.file(ilk, what, data) **X** a, 670
 ManagedGemJoin.constructor(vat_, ilk_, gem_) **X**, 449

definition

AuthGemJoin5.dec: uint256, 503
 AuthGemJoin8.dec: uint256, 505
 AuthGemJoin.dec: uint256, 447, 507
 GemJoin2.dec: uint256, 427
 GemJoin3.dec: uint256, 429
 GemJoin4.dec: uint256, 433
 GemJoin5.dec: uint256, 435
 GemJoin6.dec: uint256, 437
 GemJoin7.dec: uint256, 439
 GemJoin8.dec: uint256, 441
 GemJoin9.dec: uint256, 443
 GemJoin.dec: uint256, 423
 IlkRegistry.dec(ilk), 672
 ManagedGemJoin.dec: uint256, 449

read

AuthGemJoin5.constructor(vat_, ilk_, gem_) **X**, 503
 AuthGemJoin5.exit(usr, amt) **X**, 504
 AuthGemJoin5.join(urn, amt, msgSender) **X** a, 504
 AuthGemJoin8.constructor(vat_, ilk_, gem_) **X**, 505
 AuthGemJoin8.exit(usr, amt) **X**, 506
 AuthGemJoin8.join(urn, amt, msgSender) **X** a, 506
 GemJoin3.exit(usr, amt) **X**, 430
 GemJoin3.join(usr, amt) **X**, 430
 GemJoin5.constructor(vat_, ilk_, gem_) **X**, 435
 GemJoin5.exit(usr, amt) **X**, 436
 GemJoin5.join(usr, amt) **X**, 436
 GemJoin7.constructor(vat_, ilk_, gem_) **X**, 439
 GemJoin7.exit(usr, amt) **X**, 440
 GemJoin7.join(usr, amt) **X**, 440
 GemJoin8.constructor(vat_, ilk_, gem_) **X**, 441
 GemJoin8.exit(usr, amt) **X**, 442
 GemJoin8.join(usr, amt) **X**, 442
 IlkRegistry.dec(ilk), 672
 ManagedGemJoin.exit(urn, usr, amt) **X** a, 450
 ManagedGemJoin.join(usr, amt) **X** a, 450

decimals

call

AuthGemJoin5.constructor(vat_, ilk_, gem_) X, 503
 AuthGemJoin8.constructor(vat_, ilk_, gem_) X, 505
 AuthGemJoin.constructor(vat_, ilk_, gem_) X, 447, 507
 D3MInit.init(dss, d3m, cfg, gem), 725
 DssExecLib.addCollateralBase(_ilk, _gem, _join, _clip, _calc, _pip) X, 749
 GemJoin2.constructor(vat_, ilk_, gem_) X, 427
 GemJoin4.constructor(vat_, ilk_, gem_) X, 433
 GemJoin5.constructor(vat_, ilk_, gem_) X, 435
 GemJoin6.constructor(vat_, ilk_, gem_) X, 437
 GemJoin7.constructor(vat_, ilk_, gem_) X, 439
 GemJoin8.constructor(vat_, ilk_, gem_) X, 441
 GemJoin9.constructor(vat_, ilk_, gem_) X, 443
 GemJoin.constructor(vat_, ilk_, gem_) X, 424
 ManagedGemJoin.constructor(vat_, ilk_, gem_) X, 449

definition

AAVE.decimals: uint8, 451
 BAL.decimals: uint8, 453
 BAT.decimals: uint256, 455
 COMP.decimals: uint8, 457
 Dai.decimals: uint8, 223, 587, 613
 DAIMock.decimals() [ERC20], 627
 GNT.decimals: uint256, 461
 GUSD.decimals: uint8, 463
 KNC.decimals: uint256, 465
 LINK.decimals: uint8, 467
 LRC.decimals: uint8, 469
 MANA.decimals: uint8, 471
 MATIC.decimals: uint256, 473
 OMG.decimals: uint256, 475
 PAXG.decimals: uint256, 477
 PAXUSD.decimals: uint256, 481
 RENBTC.decimals: uint256, 483
 REP.decimals: uint256, 485
 TUSD.decimals: uint8, 487
 UNI.decimals: uint8, 489
 USDC.decimals: uint256, 491
 USDT.decimals: uint256, 493
 WBTC.decimals: uint256, 495
 WETH9_.decimals: uint8, 517
 WSTETH.decimals: uint8, 497
 YFI.decimals: uint8, 499
 ZRX.decimals: uint256, 501

decreaseAllowance

definition

Dai.decreaseAllowance(spender, subtractedValue) X, 589, 615
 DAIMock.decreaseAllowance(spender, subtractedValue) [ERC20] X, 629

decreaseGlobalDebtCeiling

call

DssExecLib.decreaseIlkDebtCeiling(_ilk, _amount, _global) X, 742

definition

DssExecLib.decreaseGlobalDebtCeiling(_amount) X, 737

decreaseIlkDebtCeiling

definition

DssExecLib.decreaseIlkDebtCeiling(_ilk, _amount, _global) X, 742

DEFAULT_DELIMITER

definition

ScriptTools.DEFAULT_DELIMITER: string, 713

read

ScriptTools.readAddressArray(json, key, envKey), 718
 ScriptTools.readBoolArray(json, key, envKey), 718
 ScriptTools.readBytes32Array(json, key, envKey), 717

ScriptTools.readByteArray(json, key, envKey), 718
 ScriptTools.readIntArray(json, key, envKey), 717
 ScriptTools.readStringArray(json, key, envKey), 717
 ScriptTools.readUIntArray(json, key, envKey), 717

defaultRpcUrls

write

D3MCoreDeployScript.setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains], 346
 D3MCoreInitScript.setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains], 374
 D3MDeployScript.setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains], 400
 D3MInitScript.setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains], 420

definition

D3MCoreDeployScript.defaultRpcUrls [StdChains] : mapping(string => string), 330
 D3MCoreInitScript.defaultRpcUrls [StdChains] : mapping(string => string), 358
 D3MDeployScript.defaultRpcUrls [StdChains] : mapping(string => string), 384
 D3MInitScript.defaultRpcUrls [StdChains] : mapping(string => string), 404

read

D3MCoreDeployScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 345
 D3MCoreInitScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 373
 D3MDeployScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 399
 D3MInitScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 419

DEFAULT_SENDER

definition

D3MCoreDeployScript.DEFAULT_SENDER [CommonBase] : address, 329
 D3MCoreInitScript.DEFAULT_SENDER [CommonBase] : address, 357
 D3MDeployScript.DEFAULT_SENDER [CommonBase] : address, 383
 D3MInitScript.DEFAULT_SENDER [CommonBase] : address, 403

DEFAULT_TEST_CONTRACT

definition

D3MCoreDeployScript.DEFAULT_TEST_CONTRACT [CommonBase] : address, 329
 D3MCoreInitScript.DEFAULT_TEST_CONTRACT [CommonBase] : address, 357
 D3MDeployScript.DEFAULT_TEST_CONTRACT [CommonBase] : address, 383
 D3MInitScript.DEFAULT_TEST_CONTRACT [CommonBase] : address, 403

delay

call

DssExec.schedule() X, 309

delegate

call

D3MInit.initCompound(dss, d3m, cfg, compoundCfg), 727
 D3MInitScript.run() X, 420

delegates

write

D3MCompoundPlan.constructor(cDai_) X, 315
 D3MCompoundPlan.file(what, addr, data) X a, 316

definition

D3MCompoundPlan.delegates: mapping(address => uint256), 315

read

D3MCompoundPlan.active(), 317

delegateVat

definition

DssExecLib.delegateVat(_usr) X, 734

DELIMITER_OVERRIDE

definition

ScriptTools.DELIMITER_OVERRIDE: string, 713

read

ScriptTools.readAddressArray(json, key, envKey), 718
 ScriptTools.readBoolArray(json, key, envKey), 718
 ScriptTools.readBytes32Array(json, key, envKey), 717
 ScriptTools.readByteArray(json, key, envKey), 718
 ScriptTools.readIntArray(json, key, envKey), 717
 ScriptTools.readStringArray(json, key, envKey), 717
 ScriptTools.readUIntArray(json, key, envKey), 717

dent

definition

Flopper.dent(id, lot, bid) [X](#), 267

deny

 call

 DssExecLib.deauthorize(_base, _ward) [X](#), 734

 ESM.denyProxy(target) [X](#), 305

 ESM.fire() [X](#), 304

 ScriptTools.switchOwner(base, deployer, newOwner), 716

 definition

 AuthGemJoin5.deny(usr) [X](#) [a](#), 503

 AuthGemJoin8.deny(usr) [X](#) [a](#), 505

 AuthGemJoin.deny(usr) [X](#) [a](#), 447, 507

 BasicRelay.deny(usr) [X](#) [a](#), 657

 ChainLog.deny(usr) [X](#) [a](#), 709

 Clipper.deny(usr) [X](#) [a](#), 242

 Cure.deny(usr) [X](#) [a](#), 292

 D3MAavePlan.deny(usr) [X](#) [a](#), 312

 D3MAavePool.deny(usr) [X](#) [a](#), 350

 D3MCompoundPlan.deny(usr) [X](#) [a](#), 316

 D3MCompoundPool.deny(usr) [X](#) [a](#), 378

 D3MHub.deny(usr) [X](#) [a](#), 321

 D3MOracle.deny(usr) [X](#) [a](#), 353

 Dai.deny(guy) [X](#) [a](#), 223

 Dai.deny(usr) [X](#) [a](#), 587, 613

 DaiJoin.deny(usr) [X](#) [a](#), 425

 Dog.deny(usr) [X](#) [a](#), 252

 DssAutoLine.deny(usr) [X](#) [a](#), 707

 DssPsm.deny(usr) [X](#) [a](#), 509

 DssVestMintable.deny(_usr) [DssVest] [X](#) [a](#), 677

 DssVestSuckable.deny(_usr) [DssVest] [X](#) [a](#), 687

 DssVestTransferrable.deny(_usr) [DssVest] [X](#) [a](#), 697

 End.deny(usr) [X](#) [a](#), 297

 ESM.deny(usr) [X](#) [a](#), 304

 ExponentialDecrease.deny(usr) [X](#) [a](#), 239

 Flapper.deny(usr) [X](#) [a](#), 262

 Flopper.deny(usr) [X](#) [a](#), 266

 GemJoin2.deny(usr) [X](#) [a](#), 427

 GemJoin3.deny(usr) [X](#) [a](#), 429

 GemJoin4.deny(usr) [X](#) [a](#), 433

 GemJoin5.deny(usr) [X](#) [a](#), 435

 GemJoin6.deny(usr) [X](#) [a](#), 437

 GemJoin7.deny(usr) [X](#) [a](#), 439

 GemJoin8.deny(usr) [X](#) [a](#), 441

 GemJoin9.deny(usr) [X](#) [a](#), 443

 GemJoin.deny(usr) [X](#) [a](#), 424

 IlkRegistry.deny(usr) [X](#) [a](#), 667

 Jug.deny(usr) [X](#) [a](#), 273

 L1DAIBridge.deny(usr) [X](#) [a](#), 618

 L1DaiGateway.deny(usr) [X](#) [a](#), 571

 L1DAITokenBridge.deny(usr) [X](#) [a](#), 593

 L1Escrow.deny(usr) [X](#) [a](#), 575, 597, 635

 L1GovernanceRelay.deny(usr) [X](#) [a](#), 578, 601, 621

 L2DaiGateway.deny(usr) [X](#) [a](#), 582

 L2DAITokenBridge.deny(usr) [X](#) [a](#), 605

 LinearDecrease.deny(usr) [X](#) [a](#), 235

 ManagedGemJoin.deny(usr) [X](#) [a](#), 449

 Median.deny(usr) [X](#) [a](#), 282

 OSM.deny(usr) [X](#) [a](#), 286

 Pot.deny(guy) [X](#) [a](#), 277

 Spotter.deny(guy) [X](#) [a](#), 227

 StairstepExponentialDecrease.deny(usr) [X](#) [a](#), 237

 TeleportJoin.deny(usr) [X](#) [a](#), 642

 TeleportOracleAuth.deny(usr) [X](#) [a](#), 649

 TeleportRouter.deny(usr) [X](#) [a](#), 653

TrustedRelay.deny(usr) [X](#) [a](#), 662
 Vat.deny(usr) [X](#) [a](#), 229
 Vow.deny(usr) [X](#) [a](#), 269

denyProxy

- definition
 - ESM.denyProxy(target) [X](#), 305

dependencies

- write
 - D3MCoreInitScript.run() [X](#), 374
 - D3MDeployScript.run() [X](#), 400
 - D3MInitScript.run() [X](#), 420
- definition
 - D3MCoreInitScript.dependencies: [string](#), 357
 - D3MDeployScript.dependencies: [string](#), 383
 - D3MInitScript.dependencies: [string](#), 403
- read
 - D3MCoreInitScript.run() [X](#), 374
 - D3MDeployScript.run() [X](#), 400
 - D3MInitScript.run() [X](#), 420

deployAave

- call
 - D3MDeployScript.run() [X](#), 400
- definition
 - D3MDeploy.deployAave(deployer, owner, ilk, vat, hub, dai, lendingPool), 723

deployCode

- definition
 - D3MCoreDeployScript.deployCode(what) [StdCheatsSafe], 342
 - D3MCoreDeployScript.deployCode(what, args) [StdCheatsSafe], 341
 - D3MCoreDeployScript.deployCode(what, args, val) [StdCheatsSafe], 342
 - D3MCoreDeployScript.deployCode(what, val) [StdCheatsSafe], 342
 - D3MCoreInitScript.deployCode(what) [StdCheatsSafe], 370
 - D3MCoreInitScript.deployCode(what, args) [StdCheatsSafe], 369
 - D3MCoreInitScript.deployCode(what, args, val) [StdCheatsSafe], 370
 - D3MCoreInitScript.deployCode(what, val) [StdCheatsSafe], 370
 - D3MDeployScript.deployCode(what) [StdCheatsSafe], 396
 - D3MDeployScript.deployCode(what, args) [StdCheatsSafe], 395
 - D3MDeployScript.deployCode(what, args, val) [StdCheatsSafe], 396
 - D3MDeployScript.deployCode(what, val) [StdCheatsSafe], 396
 - D3MInitScript.deployCode(what) [StdCheatsSafe], 416
 - D3MInitScript.deployCode(what, args) [StdCheatsSafe], 416
 - D3MInitScript.deployCode(what, args, val) [StdCheatsSafe], 416
 - D3MInitScript.deployCode(what, val) [StdCheatsSafe], 416

deployCompound

- call
 - D3MDeployScript.run() [X](#), 400
- definition
 - D3MDeploy.deployCompound(deployer, owner, ilk, vat, hub, cdai), 723

deployCore

- call
 - D3MCoreDeployScript.run() [X](#), 346
- definition
 - D3MDeploy.deployCore(deployer, owner, daiJoin), 723

deploymentChainId

- write
 - Dai.constructor() [X](#), 588, 614
- definition
 - Dai.deploymentChainId: [uint256](#), 587, 613
- read
 - Dai.DOMAIN_SEPARATOR(), 588, 614
 - Dai.permit(owner, spender, [value](#), deadline, v, r, s) [X](#), 590, 616

DEPOSIT

- definition
 - L1DAIBridge.DEPOSIT: [uint256](#), 617

read
 L1DAIBridge.cancelDeposit(amount, l2Recipient, l1Recipient, nonce) X, 620
 L1DAIBridge.deposit(amount, l2Recipient) X, 619
 L1DAIBridge.startDepositCancellation(amount, l2Recipient, nonce) X, 620

deposit
 call
 CurveLpTokenUniv3Callee.clipperCall(sender, owe, slice, data) X, 552
 D3MAavePool.deposit(wad) X, 350
 D3MHub._exec(ilk, _pool, Art, lineWad), 322
 DssProxyActions.ethJoin_join(apt, urn) X, 531
 WETH9._fallback() X, 517
 WstETHCurveUniv3Callee.clipperCall(sender, owe, slice, data) X, 564
 definition
 D3MAavePool.deposit(wad) X, 350
 D3MCompoundPool.deposit(wad) X, 378
 L1DAIBridge.deposit(amount, l2Recipient) X, 619
 WETH9._deposit() X, 517

depositERC20
 definition
 L1DAITokenBridge.depositERC20(_l1Token, _l2Token, _amount, _l2Gas, _data) X, 594

depositERC20To
 definition
 L1DAITokenBridge.depositERC20To(_l1Token, _l2Token, _to, _amount, _l2Gas, _data) X, 594

deprecate
 definition
 USDT.deprecate(_upgradedAddress) X, 494

deprecated
 write
 USDT.deprecate(_upgradedAddress) X, 494
 definition
 USDT.deprecated: bool, 493

deriveKey
 call
 D3MCoreDeployScript.deriveRememberKey(mnemonic, index) [StdCheatsSafe], 343
 D3MCoreInitScript.deriveRememberKey(mnemonic, index) [StdCheatsSafe], 371
 D3MDeployScript.deriveRememberKey(mnemonic, index) [StdCheatsSafe], 397
 D3MInitScript.deriveRememberKey(mnemonic, index) [StdCheatsSafe], 417

deriveRememberKey
 definition
 D3MCoreDeployScript.deriveRememberKey(mnemonic, index) [StdCheatsSafe], 343
 D3MCoreInitScript.deriveRememberKey(mnemonic, index) [StdCheatsSafe], 371
 D3MDeployScript.deriveRememberKey(mnemonic, index) [StdCheatsSafe], 397
 D3MInitScript.deriveRememberKey(mnemonic, index) [StdCheatsSafe], 417

description
 call
 DssExec.description(), 309
 definition
 DssExec.description(), 309

DGD
 definition
 contract DGD, 459

_diff
 call
 Jug.drip(ilk) X, 275
 definition
 Jug._diff(x, y), 274

digs
 call
 Clipper.take(id, amt, max, who, data) X, 246
 Clipper.yank(id) X a, 249
 definition
 Dog.digs(ilk, rad) X a, 255

Dirt

write

Dog.bark(ilk, urn, kpr) [X](#), 253

Dog.digs(ilk, rad) [X](#) [a](#), 255

definition

Dog.Dirt: [uint256](#), 251

read

Dog.bark(ilk, urn, kpr) [X](#), 253

Dog.digs(ilk, rad) [X](#) [a](#), 255

dirt

write

Dog.bark(ilk, urn, kpr) [X](#), 253

Dog.digs(ilk, rad) [X](#) [a](#), 255

read

Dog.digs(ilk, rad) [X](#) [a](#), 255

disable

call

D3MMom.disable(who) [X](#) [a](#), 356

definition

D3MAavePlan.disable() [X](#), 314

D3MCompoundPlan.disable() [X](#), 317

D3MMom.disable(who) [X](#) [a](#), 356

diss

call

DssExecLib.removeReaderFromWhitelist(_oracle, _reader) [X](#), 747

definition

Median.diss(a) [X](#) [a](#), 284

OSM.diss(a) [X](#) [a](#), 288

TrustedRelay.diss(usr) [X](#) [a](#), 662

div

call

PAXG.getFeeFor(_value), 479

USDT.transfer(_to, _value) [X](#), 493

USDT.transferFrom(_from, _to, _value) [X](#), 494

definition

PAXG.div(a, b), 477

SafeMath.div(a, b), 753

_divup

call

CalleeMakerOtcDai.clipperCall(sender, daiAmt, gemAmt, data) [X](#), 550

CurveLpTokenUniv3Callee.clipperCall(sender, owe, slice, data) [X](#), 552

D3MHub._exec(ilk, _pool, Art, lineWad), 322

PSMCallee.clipperCall(sender, owe, slice, data) [X](#), 555

UniswapV2CalleeDai.clipperCall(sender, daiAmt, gemAmt, data) [X](#), 558

UniswapV2LpTokenCalleeDai.clipperCall(sender, daiAmt, gemAmt, data) [X](#), 560

UniswapV3Callee.clipperCall(sender, owe, slice, data) [X](#), 567

WstETHCurveUniv3Callee.clipperCall(sender, owe, slice, data) [X](#), 564

definition

CalleeMakerOtcDai.divup(x, y) [CalleeMakerOtc], 549

CurveLpTokenUniv3Callee._divup(x, y), 551

D3MHub._divup(x, y), 320

PSMCallee._divup(x, y), 555

UniswapV2CalleeDai.divup(x, y) [UniswapV2Callee], 557

UniswapV2LpTokenCalleeDai.divup(x, y) [UniswapV2Callee(2)], 559

UniswapV3Callee._divup(x, y), 567

WstETHCurveUniv3Callee._divup(x, y), 563

Dog

definition

[contract](#) Dog, 251

dog

call

DssExecLib.addCollateralBase(ilk, _gem, _join, _clip, _calc, _pip) [X](#), 749

DssExecLib.setIlkLiquidationPenalty(ilk, _pct_bps) [X](#), 743

```

DssExecLib.setIlkMaxLiquidationAmount(_ilk, _amount) X, 744
DssExecLib.setIlkMinVaultAmount(_ilk, _amount) X, 743
DssExecLib.setMaxTotalDAILiquidationAmount(_amount) X, 740
write
  Clipper.constructor(vat_, spotter_, dog_, ilk_) X, 243
  Clipper.file(what, data) X a, 243
  End.file(what, data) X a, 298
  IlkRegistry.constructor(vat_, dog_, cat_, spot_) X, 668
  IlkRegistry.file(what, data) X a, 669
definition
  Clipper.dog: DogLike, 241
  DssExecLib.dog(), 730
  End.dog: DogLike, 295
  IlkRegistry.dog: DogLike, 667
read
  Clipper.take(id, amt, max, who, data) X, 246
  Clipper.upchost() X, 248
  Clipper.yank(id) X a, 249
  End.cage() X a, 299
  End.snip(ilk, id) X, 299
  IlkRegistry.add(adapter) X, 668
  IlkRegistry.constructor(vat_, dog_, cat_, spot_) X, 668
domain
  write
    TeleportJoin.constructor(vat_, daiJoin_, ilk_, domain_) X, 641
  definition
    TeleportJoin.domain: bytes32, 641
  read
    TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643
domainAt
  definition
    TeleportRouter.domainAt(index), 654
domains
  write
    TeleportRouter.file(what, domain, data) X a, 653
  definition
    TeleportRouter.domains: mapping(address => bytes32), 653
  read
    TeleportRouter.settle(targetDomain, batchedDaiToFlush) X, 655
_DOMAIN_SEPARATOR
  write
    Dai.constructor() X, 588, 614
    Dai.constructor(chainId_) X, 224
  definition
    Dai.DOMAIN_SEPARATOR(), 588, 614
    Dai.DOMAIN_SEPARATOR: bytes32, 223, 587, 613
  read
    Dai.DOMAIN_SEPARATOR(), 588, 614
    Dai.permit(holder, spender, nonce, expiry, allowed, v, r, s) X, 225
    Dai.permit(owner, spender, value, deadline, v, r, s) X, 590, 616
done
  write
    DssExec.cast() X, 310
  definition
    DssExec.done: bool, 309
  read
    DssExec.cast() X, 310
draw
  definition
    DssProxyActions.draw(manager, jug, daiJoin, cdp, wad) X, 536
drip
  call
    DssExecLib.accumulateCollateralStabilityFees(_ilk) X, 735

```

DssExecLib.accumulateDSR() [X](#), 735
 DssExecLib.setDSR(_rate, _doDrip) [X](#), 737
 DssExecLib.setIlkStabilityFee(_ilk, _rate, _doDrip) [X](#), 746
 DssProxyActionsDsr.exit(daiJoin, pot, wad) [X](#), 548
 DssProxyActionsDsr.exitAll(daiJoin, pot) [X](#), 548
 DssProxyActionsDsr.join(daiJoin, pot, wad) [X](#), 547
 DssProxyActions._getDrawDart(vat, jug, urn, ilk, wad), 530
 definition
 Jug.drip(ilk) [X](#), 275
 Pot.drip() [X](#), 279
 drop
 definition
 Cure.drop(src) [X](#) [a](#), 293
 Median.drop(a) [X](#) [a](#), 283
 dsr
 write
 Pot.constructor(vat_) [X](#), 277
 Pot.cage() [X](#) [a](#), 279
 Pot.file(what, data) [X](#) [a](#), 278
 definition
 Pot.dsr: [uint256](#), 277
 read
 Pot.drip() [X](#), 279
 dss
 write
 D3MCoreDeployScript.run() [X](#), 346
 D3MCoreInitScript.run() [X](#), 374
 D3MDeployScript.run() [X](#), 400
 D3MInitScript.run() [X](#), 420
 definition
 D3MCoreDeployScript.dss: DssInstance, 329
 D3MCoreInitScript.dss: DssInstance, 357
 D3MDeployScript.dss: DssInstance, 383
 D3MInitScript.dss: DssInstance, 403
 read
 D3MCoreDeployScript.run() [X](#), 346
 D3MCoreInitScript.run() [X](#), 374
 D3MDeployScript.run() [X](#), 400
 D3MInitScript.run() [X](#), 420
 DssAction
 definition
 contract DssAction, 307
 DssAutoLine
 definition
 contract DssAutoLine, 705
 DssCdpManager
 definition
 contract DssCdpManager, 521
 DssExec
 definition
 contract DssExec, 309
 DssExecLib
 definition
 library DssExecLib, 729
 DssProxyActions
 definition
 contract DssProxyActions, 529
 DssProxyActionsDsr
 definition
 contract DssProxyActionsDsr, 547
 DssProxyActionsEnd
 definition
 contract DssProxyActionsEnd, 543

DssPsm
 [definition](#)
 [contract](#) DssPsm, 509

DssVestMintable
 [definition](#)
 [contract](#) DssVestMintable, 675

DssVestSuckable
 [definition](#)
 [contract](#) DssVestSuckable, 685

DssVestTransferrable
 [definition](#)
 [contract](#) DssVestTransferrable, 695

DSWethFactory
 [definition](#)
 [contract](#) DSWethFactory, 513

dump
 [write](#)
 Vow.file(what, [data](#)) [X](#) [a](#), 270
 [definition](#)
 Vow.dump: [uint256](#), 269
 [read](#)
 Vow.flop() [X](#), 271

dust
 [write](#)
 Vat.file(ilk, what, [data](#)) [X](#) [a](#), 231

duty
 [write](#)
 Jug.file(ilk, what, [data](#)) [X](#) [a](#), 275
 [read](#)
 Jug.drip(ilk) [X](#), 275

E

EIP1559ScriptArtifact

definition

`struct` StdCheatsSafe.EIP1559ScriptArtifact, 333, 361, 387, 407

either

call

Vat.fork(ilk, src, dst, dink, dart) **X**, 233Vat.frob(i, u, v, w, dink, dart) **X**, 232

Vat.wish(bit, usr), 230

definition

Vat.either(x, y), 232

End

definition

`contract` End, 295

end

call

D3MAavePool.exit(dst, wad) **X**, 351D3MCompoundPool.exit(dst, wad) **X**, 379DssExecLib.addCollateralBase(_ilk, _gem, _join, _clip, _calc, _pip) **X**, 749DssExecLib.addNewCollateral(co) **X**, 750DssExecLib.setEmergencyShutdownProcessingTime(_duration) **X**, 740

write

D3MHub.file(what, data) **X** a, 321ESM.constructor(gem_, end_, proxy_, min_) **X**, 303ESM.file(what, data) **X** a, 304Flapper.kick(lot, bid) **X** a, 263Flapper.tick(id) **X**, 263Flopper.kick(gal, lot, bid) **X** a, 266Flopper.tick(id) **X**, 267

definition

D3MHub.end: EndLike, 319

DssExecLib.end(), 731

ESM.end: EndLike, 303

read

D3MHub.exec(ilk) **X**, 324ESM.fire() **X**, 304ESM.join(wad) **X**, 305Flapper.deal(id) **X**, 264Flapper.tend(id, lot, bid) **X**, 263Flapper.tick(id) **X**, 263Flopper.deal(id) **X**, 267Flopper.dent(id, lot, bid) **X**, 267Flopper.tick(id) **X**, 267

enter

call

DssProxyActions.enter(manager, src, cdp) **X**, 534

definition

DssCdpManager.enter(src, cdp) **X**, 525DssProxyActions.enter(manager, src, cdp) **X**, 534

EnumerableSet

definition

`library` EnumerableSet, 755

envOr

call

D3MCoreDeployScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 345

D3MCoreInitScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 373

D3MDeployScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 399

D3MInitScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 419

ScriptTools.exportContract(name, label, addr), 715

ScriptTools.loadConfig(), 714

ScriptTools.loadConfig(name), 714

ScriptTools.loadDependencies(), 715

ScriptTools.loadDependencies(name), 714

```

ScriptTools.readAddress(json, key, envKey), 717
ScriptTools.readAddressArray(json, key, envKey), 718
ScriptTools.readBool(json, key, envKey), 718
ScriptTools.readBoolArray(json, key, envKey), 718
ScriptTools.readBytes(json, key, envKey), 718
ScriptTools.readBytes32(json, key, envKey), 717
ScriptTools.readBytes32Array(json, key, envKey), 717
ScriptTools.readByteArray(json, key, envKey), 718
ScriptTools.readInt(json, key, envKey), 717
ScriptTools.readIntArray(json, key, envKey), 717
ScriptTools.readString(json, key, envKey), 717
ScriptTools.readStringArray(json, key, envKey), 717
ScriptTools.readUint(json, key, envKey), 716
ScriptTools.readUintArray(json, key, envKey), 717
envString
  call
    ScriptTools.exportContract(label, addr), 715
    ScriptTools.loadConfig(), 714
    ScriptTools.loadDependencies(), 715
envUint
  call
    ScriptTools.getRootChainId(), 713
eq
  call
    D3MDeployScript.run() X, 400
    D3MInitScript.run() X, 420
    ScriptTools.loadConfig(), 714
    ScriptTools.loadConfig(name), 714
    ScriptTools.loadDependencies(), 715
    ScriptTools.loadDependencies(name), 714
  definition
    ScriptTools.eq(a, b), 716
era
  call
    OSM.pass(), 287
    OSM.poke() X, 287
  definition
    OSM.era(), 287
erc20Impl
  call
    AuthGemJoin8.constructor(vat_, ilk_, gem_) X, 505
    AuthGemJoin8.exit(usr, amt) X, 506
    AuthGemJoin8.join(urn, amt, msgSender) X a, 506
    GemJoin8.constructor(vat_, ilk_, gem_) X, 441
    GemJoin8.exit(usr, amt) X, 442
    GemJoin8.join(usr, amt) X, 442
  write
    GUSD.setImplementation(newImplementation) X, 463
  definition
    GUSD.erc20Impl: address, 463
escrow
  write
    L1DAIBridge.constructor(_starkNet, _dai, _l2Dai, _escrow, _l2DaiBridge) X, 618
    L1DAITokenBridge.constructor(_l1Token, _l2DAITokenBridge, _l2Token, _l1messenger,
      _escrow) X, 593
    L1EscrowMom.constructor(escrow_, token_) X, 637
  definition
    L1DAIBridge.escrow: address, 617
    L1DAITokenBridge.escrow: address, 591
    L1EscrowMom.escrow: address, 637
read
    L1DAIBridge.cancelDeposit(amount, l2Recipient, l1Recipient, nonce) X, 620
    L1DAIBridge.deposit(amount, l2Recipient) X, 619

```


L1DAIBridge.withdraw(amount, l1Recipient) **X**, 619
 L1DAITokenBridge.finalizeERC20Withdrawal(_l1Token, _l2Token, _from, _to, _amount, _data) **X**, 595
 L1DAITokenBridge._initiateERC20Deposit(_from, _to, _amount, _l2Gas, _data), 594
 L1EscrowMom.refuse(spender) **X a**, 638

ESM

definition

contract ESM, 303

esm

call

DssExecLib.addCollateralBase(_ilk, _gem, _join, _clip, _calc, _pip) **X**, 749

definition

DssExecLib.esm(), 731

eta

write

DssExec.schedule() **X**, 309

definition

DssExec.eta: uint256, 309

read

DssExec.cast() **X**, 310
 DssExec.nextCastTime(), 309
 DssExec.schedule() **X**, 309

ETH

definition

CurveLpTokenUniv3Callee.ETH: address, 551

read

CurveLpTokenUniv3Callee.clipperCall(sender, owe, slice, data) **X**, 552

ethJoin_join

call

DssProxyActions.lockETH(manager, ethJoin, cdp) **X**, 534
 DssProxyActions.lockETHAndDraw(manager, jug, ethJoin, daiJoin, cdp, wadD) **X**, 538

definition

DssProxyActions.ethJoin_join(apt, urn) **X**, 531

ethPriceOracle

write

TrustedRelay.constructor(_oracleAuth, _daiJoin, _ethPriceOracle) **X**, 661

definition

TrustedRelay.ethPriceOracle: DsValueLike, 661

read

TrustedRelay.relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s) **X**, 663

exactInput

call

CurveLpTokenUniv3Callee.clipperCall(sender, owe, slice, data) **X**, 552
 UniswapV3Callee.clipperCall(sender, owe, slice, data) **X**, 567
 WstETHCurveUniv3Callee.clipperCall(sender, owe, slice, data) **X**, 564

EXCESS_UTILIZATION_RATE

call

D3MAavePlan._calculateTargetSupply(targetInterestRate, totalDebt), 313

exchange

call

WstETHCurveUniv3Callee.clipperCall(sender, owe, slice, data) **X**, 564

exchangeRateStored

call

D3MCompoundPool.deposit(wad) **X**, 378

exec

call

D3MHub.exec(ilk) **X**, 324
 DssExec.cast() **X**, 310

definition

D3MHub.exec(ilk) **X**, 324
 D3MHub._exec(ilk, _pool, Art, lineWad), 322
 DssAutoLine.exec(_ilk) **X**, 707

execute

definition

DssAction.execute() [X](#), 307

exit

call

CalleeMakerOtcDai.clipperCall([sender](#), daiAmt, gemAmt, [data](#)) [X](#), 550

CurveLpTokenUniv3Callee.clipperCall([sender](#), owe, slice, [data](#)) [X](#), 552

D3MHub._exec(ilk, _pool, Art, lineWad), 322

D3MHub.exit(ilk, usr, wad) [X](#), 325

DssExecLib.sendPaymentFromSurplusBuffer(_target, _amount) [X](#), 751

DssProxyActions.draw(manager, jug, daiJoin, cdp, wad) [X](#), 536

DssProxyActionsDsr.exit(daiJoin, pot, wad) [X](#), 548

DssProxyActionsDsr.exitAll(daiJoin, pot) [X](#), 548

DssProxyActionsEnd.cashETH(ethJoin, end, ilk, wad) [X](#), 544

DssProxyActionsEnd.cashGem(gemJoin, end, ilk, wad) [X](#), 545

DssProxyActionsEnd.freeETH(manager, ethJoin, end, cdp) [X](#), 544

DssProxyActionsEnd.freeGem(manager, gemJoin, end, cdp) [X](#), 544

DssProxyActions.exitETH(manager, ethJoin, cdp, wad) [X](#), 536

DssProxyActions.exitGem(manager, gemJoin, cdp, amt) [X](#), 536

DssProxyActions.freeETH(manager, ethJoin, cdp, wad) [X](#), 535

DssProxyActions.freeGem(manager, gemJoin, cdp, amt) [X](#), 535

DssProxyActions.lockETHAndDraw(manager, jug, ethJoin, daiJoin, cdp, wadD) [X](#), 538

DssProxyActions.lockGemAndDraw(manager, jug, gemJoin, daiJoin, cdp, amtC, wadD, transferFrom) [X](#), 539

DssProxyActions.wipeAllAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC) [X](#), 540

DssProxyActions.wipeAllAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC) [X](#), 541

DssProxyActions.wipeAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC, wadD) [X](#), 540

DssProxyActions.wipeAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC, wadD) [X](#), 541

DssPsm.buyGem(usr, gemAmt) [X](#), 511

DssPsm.sellGem(usr, gemAmt) [X](#), 510

DssVestSuckable.pay(_guy, _amt), 693

GemJoin4.join(usr, wad) [X](#), 434

PSMCallee.clipperCall([sender](#), owe, slice, [data](#)) [X](#), 555

TeleportJoin._mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643

UniswapV2CalleeDai.clipperCall([sender](#), daiAmt, gemAmt, [data](#)) [X](#), 558

UniswapV2LpTokenCalleeDai.clipperCall([sender](#), daiAmt, gemAmt, [data](#)) [X](#), 560

UniswapV3Callee.clipperCall([sender](#), owe, slice, [data](#)) [X](#), 567

WstETHCurveUniv3Callee.clipperCall([sender](#), owe, slice, [data](#)) [X](#), 564

definition

AuthGemJoin5.exit(usr, amt) [X](#), 504

AuthGemJoin8.exit(usr, amt) [X](#), 506

AuthGemJoin.exit(guy, wad) [X](#), 508

AuthGemJoin.exit(usr, wad) [X](#), 448

D3MAavePool.exit(dst, wad) [X](#), 351

D3MCompoundPool.exit(dst, wad) [X](#), 379

D3MHub.exit(ilk, usr, wad) [X](#), 325

DaiJoin.exit(usr, wad) [X](#), 426

DssProxyActionsDsr.exit(daiJoin, pot, wad) [X](#), 548

GemBag.exit(usr, wad) [X](#), 431

GemJoin2.exit(usr, wad) [X](#), 428

GemJoin3.exit(usr, amt) [X](#), 430

GemJoin4.exit(usr, wad) [X](#), 434

GemJoin5.exit(usr, amt) [X](#), 436

GemJoin6.exit(usr, wad) [X](#), 438

GemJoin7.exit(usr, amt) [X](#), 440

GemJoin8.exit(usr, amt) [X](#), 442

GemJoin9.exit(usr, wad) [X](#), 445

GemJoin.exit(usr, wad) [X](#), 424

ManagedGemJoin.exit(urn, usr, amt) [X](#) [a](#), 450

Pot.exit(wad) [X](#), 279

exitAll

definition

DssProxyActionsDsr.exitAll(daiJoin, pot) [X](#), 548

exited

write

D3MAavePool.exit(dst, wad) **X**, 351

D3MCompoundPool.exit(dst, wad) **X**, 379

definition

D3MAavePool.exited: **uint256**, 349

D3MCompoundPool.exited: **uint256**, 377

read

D3MAavePool.exit(dst, wad) **X**, 351

D3MCompoundPool.exit(dst, wad) **X**, 379

exitETH

definition

DssProxyActions.exitETH(manager, ethJoin, cdp, wad) **X**, 536

exitGem

definition

DssProxyActions.exitGem(manager, gemJoin, cdp, amt) **X**, 536

expiration

write

DssExec.constructor(_expiration, _spellAction) **X**, 309

definition

DssExec.expiration: **uint256**, 309

read

DssExec.schedule() **X**, 309

ExponentialDecrease

definition

contract ExponentialDecrease, 239

exportContract

call

D3MCoreDeployScript.run() **X**, 346

D3MDeployScript.run() **X**, 400

ScriptTools.exportContract(label, addr), 715

definition

ScriptTools.exportContract(label, addr), 715

ScriptTools.exportContract(name, label, addr), 715

EXPORT_JSON_KEY

definition

ScriptTools.EXPORT_JSON_KEY: **string**, 713

read

ScriptTools.exportContract(name, label, addr), 715

F

fallback

definition

WETH9_.fallback() **X**, 517

fee

write

TeleportConstantFee.constructor(_fee, _ttl) **X**, 639

TeleportLinearFee.constructor(_fee, _ttl) **X**, 647

definition

TeleportConstantFee.fee: **uint256**, 639

TeleportLinearFee.fee: **uint256**, 647

read

TeleportConstantFee.getFee(guid, , , , amtToTake), 639

TeleportLinearFee.getFee(guid, , , , amtToTake), 647

Feed

definition

struct OSM.Feed, 285

feeParts

definition

PAXG.feeParts: **uint256**, 477

read

PAXG.getFeeFor(_value), 479

PAXG.setFeeRate(_newFeeRate), 479

feeRate

write

PAXG.setFeeRate(_newFeeRate), 479

definition

PAXG.feeRate: **uint256**, 477

read

PAXG.getFeeFor(_value), 479

PAXG.setFeeRate(_newFeeRate), 479

feeRecipient

definition

PAXG.feeRecipient: **address**, 477

read

PAXG.transferFrom(src, dst, wad) **X**, 478

fees

write

TeleportJoin.file(what, domain_, **data**) **X** **a**, 642

definition

TeleportJoin.fees: **mapping(bytes32 => address)**, 641

read

TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643

fess

call

Dog.bark(ilk, urn, kpr) **X**, 253

definition

Vow.fess(tab) **X** **a**, 270

file

call

ClipperMom.setBreaker(clip, level, delay) **X** **a**, 258

ClipperMom.tripBreaker(clip) **X**, 259

D3MInit.init(dss, d3m, cfg, gem), 725

D3MInit.initAave(dss, d3m, cfg, aaveCfg), 726

D3MInit.initCompound(dss, d3m, cfg, compoundCfg), 727

D3MInit.initCore(dss, d3mCore), 725

DssAutoLine.exec(_ilk) **X**, 707

DssExecLib.setContract(_base, _ilk, _what, _addr) **X**, 736

DssExecLib.setContract(_base, _what, _addr) **X**, 736

DssExecLib.setValue(_base, _ilk, _what, _amt) **X**, 736

DssExecLib.setValue(_base, _what, _amt) **X**, 736

MCD.initIlk(dss, ilk, join, pip), 720

Spotter.poke(ilk) **X**, 228

definition

Clipper.file(what, data) X a, 243
 Cure.file(what, data) X a, 292
 D3MAavePlan.file(what, data) X a, 312
 D3MAavePool.file(what, data) X a, 350
 D3MCompoundPlan.file(what, addr, data) X a, 316
 D3MCompoundPlan.file(what, data) X a, 316
 D3MCompoundPool.file(what, data) X a, 378
 D3MHub.file(ilk, what, data) X a, 321
 D3MHub.file(what, data) X a, 321
 D3MOracle.file(what, data) X a, 354
 Dog.file(ilk, what, clip) X a, 253
 Dog.file(ilk, what, data) X a, 253
 Dog.file(what, data) X a, 252
 DssPsm.file(what, data) X a, 510
 DssVestMintable.file(what, data) [DssVest] X a, 677
 DssVestSuckable.file(what, data) [DssVest] X a, 687
 DssVestTransferrable.file(what, data) [DssVest] X a, 697
 End.file(what, data) X a, 298
 ESM.file(what, data) X a, 304
 ExponentialDecrease.file(what, data) X a, 239
 Flapper.file(what, data) X a, 263
 Flopper.file(what, data) X a, 266
 IlkRegistry.file(ilk, what, data) X a, 669, 670
 IlkRegistry.file(what, data) X a, 669
 Jug.file(ilk, what, data) X a, 275
 Jug.file(what, data) X a, 275
 LinearDecrease.file(what, data) X a, 236
 Pot.file(what, addr) X a, 278
 Pot.file(what, data) X a, 278
 Spotter.file(ilk, what, pip_) X a, 228
 Spotter.file(ilk, what, data) X a, 228
 Spotter.file(what, data) X a, 228
 StairstepExponentialDecrease.file(what, data) X a, 237
 TeleportJoin.file(what, domain_, data) X a, 642
 TeleportJoin.file(what, data) X a, 642
 TeleportOracleAuth.file(what, data) X a, 649
 TeleportRouter.file(what, domain, data) X a, 653
 TrustedRelay.file(what, data) X a, 662
 Vat.file(ilk, what, data) X a, 231
 Vat.file(what, data) X a, 231
 Vow.file(what, data) X a, 270

fill

write

Flapper.deal(id) X, 264
 Flapper.kick(lot, bid) X a, 263

definition

Flapper.fill: uint256, 261

read

Flapper.deal(id) X, 264
 Flapper.kick(lot, bid) X a, 263

fin

write

DssVestMintable._yank(_id, _end) [DssVest], 682
 DssVestSuckable._yank(_id, _end) [DssVest], 691
 DssVestTransferrable._yank(_id, _end) [DssVest], 701

definition

DssVestMintable.fin(_id) [DssVest], 676
 DssVestSuckable.fin(_id) [DssVest], 686
 DssVestTransferrable.fin(_id) [DssVest], 696

read

DssVestMintable.fin(_id) [DssVest], 676
 DssVestSuckable.fin(_id) [DssVest], 686

```

    DssVestTransferrable.fin(_id) [DssVest], 696
finalizeDeposit
  definition
    L2DAITokenBridge.finalizeDeposit(_l1Token, _l2Token, _from, _to, _amount, _data) X,
    606
finalizeERC20Withdrawal
  definition
    L1DAITokenBridge.finalizeERC20Withdrawal(_l1Token, _l2Token, _from, _to, _amount,
    _data) X, 595
finalizeFlush
  definition
    L1DAITeleportGateway.finalizeFlush(targetDomain, daiToFlush) X, 623
finalizeInboundTransfer
  definition
    L1DaiGateway.finalizeInboundTransfer(l1Token, from, to, amount, data) X, 572
    L2DaiGateway.finalizeInboundTransfer(l1Token, from, to, amount, ) X, 583
finalizeRegisterTeleport
  definition
    L1DAITeleportGateway.finalizeRegisterTeleport(teleport) X, 623
fire
  definition
    ESM.fire() X, 304
first
  write
    DssCdpManager.give(cdp, dst) X, 523
    DssCdpManager.open(ilk, usr) X, 522
  definition
    DssCdpManager.first: mapping(address => uint256), 521
read
    DssCdpManager.give(cdp, dst) X, 523
    DssCdpManager.open(ilk, usr) X, 522
    GetCdps.getCdpsAsc(manager, guy), 527
fix
  call
    DssProxyActionsEnd.cashETH(ethJoin, end, ilk, wad) X, 544
    DssProxyActionsEnd.cashGem(gemJoin, end, ilk, wad) X, 545
  write
    End.flow(ilk) X, 300
  definition
    End.fix: mapping(bytes32 => uint256), 295
read
    End.cash(ilk, wad) X, 301
    End.flow(ilk) X, 300
flap
  call
    DssExecLib.setMinSurplusAuctionBidIncrease(_pct_bps) X, 738
    DssExecLib.setSurplusAuctionBidDuration(_duration) X, 738
    DssExecLib.setSurplusAuctionDuration(_duration) X, 738
  definition
    DssExecLib.flap(), 731
    Vow.flap() X, 271
Flapper
  definition
    contract Flapper, 261
flapper
  write
    Vow.constructor(vat_, flapper_, flopper_) X, 269
    Vow.file(what, data) X a, 270
  definition
    Vow.flapper: FlapLike, 269
read
    Vow.cage() X a, 271
    Vow.file(what, data) X a, 270

```

Vow.flap() [X](#), 271

flip

- definition
 - DssExecLib.flip(_ilk), 732

flipperMom

- definition
 - DssExecLib.flipperMom(), 732

flog

- definition
 - Vow.flog(era) [X](#), 270

flop

- call
 - DssExecLib.setDebtAuctionBidDuration(_duration) [X](#), 739
 - DssExecLib.setDebtAuctionDuration(_duration) [X](#), 740
 - DssExecLib.setDebtAuctionMKRIncreaseRate(_pct.bps) [X](#), 740
 - DssExecLib.setMinDebtAuctionBidIncrease(_pct.bps) [X](#), 739
- definition
 - DssExecLib.flop(), 731
 - Vow.flop() [X](#), 271

Flopper

- definition
 - [contract](#) Flopper, 265

flopper

- write
 - Vow.constructor(vat_, flapper_, flopper_) [X](#), 269
 - Vow.file(what, data) [X](#) [a](#), 270
- definition
 - Vow.flopper: FlopLike, 269
- read
 - Vow.cage() [X](#) [a](#), 271
 - Vow.flop() [X](#), 271

flow

- definition
 - End.flow(ilk) [X](#), 300

flux

- call
 - Clipper.take(id, amt, max, who, data) [X](#), 246
 - Clipper.yank(id) [X](#) [a](#), 249
 - DssCdpManager.flux(cdp, dst, wad) [X](#), 524
 - DssCdpManager.flux(ilk, cdp, dst, wad) [X](#), 524
 - DssProxyActions.exitETH(manager, ethJoin, cdp, wad) [X](#), 536
 - DssProxyActions.exitGem(manager, gemJoin, cdp, amt) [X](#), 536
 - DssProxyActions.flux(manager, cdp, dst, wad) [X](#), 533
 - DssProxyActions.freeETH(manager, ethJoin, cdp, wad) [X](#), 535
 - DssProxyActions.freeGem(manager, gemJoin, cdp, amt) [X](#), 535
 - DssProxyActions.wipeAllAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC) [X](#), 540
 - DssProxyActions.wipeAllAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC) [X](#), 541
 - DssProxyActions.wipeAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC, wadD) [X](#), 540
 - DssProxyActions.wipeAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC, wadD) [X](#), 541
 - End.cash(ilk, wad) [X](#), 301
- definition
 - DssCdpManager.flux(cdp, dst, wad) [X](#), 524
 - DssCdpManager.flux(ilk, cdp, dst, wad) [X](#), 524
 - DssProxyActions.flux(manager, cdp, dst, wad) [X](#), 533
 - Vat.flux(ilk, src, dst, wad) [X](#), 231

fold

- call
 - Jug.drip(ilk) [X](#), 275
- definition
 - Vat.fold(i, u, rate) [X](#) [a](#), 234

FORCE_WITHDRAW

- definition
 - L1DAIBridge.FORCE_WITHDRAW: [uint256](#), 617

[read](#)
[L1DAIBridge.forceWithdrawal\(amount, l1Sender\)](#) [X](#), 620

[forceWithdrawal](#)
[definition](#)
[L1DAIBridge.forceWithdrawal\(amount, l1Sender\)](#) [X](#), 620

[fork](#)
[call](#)
[DssCdpManager.enter\(src, cdp\)](#) [X](#), 525
[DssCdpManager.quit\(cdp, dst\)](#) [X](#), 524
[DssCdpManager.shift\(cdpSrc, cdpDst\)](#) [X](#), 525
[definition](#)
[Vat.fork\(ilk, src, dst, dink, dart\)](#) [X](#), 233

[_free](#)
[call](#)
[DssProxyActionsEnd._free\(manager, end, cdp\)](#), 543
[DssProxyActionsEnd.freeETH\(manager, ethJoin, end, cdp\)](#) [X](#), 544
[DssProxyActionsEnd.freeGem\(manager, gemJoin, end, cdp\)](#) [X](#), 544
[definition](#)
[DssProxyActionsEnd._free\(manager, end, cdp\)](#), 543
[End.free\(ilk\)](#) [X](#), 300

[freeETH](#)
[definition](#)
[DssProxyActionsEnd.freeETH\(manager, ethJoin, end, cdp\)](#) [X](#), 544
[DssProxyActions.freeETH\(manager, ethJoin, cdp, wad\)](#) [X](#), 535

[freeGem](#)
[definition](#)
[DssProxyActionsEnd.freeGem\(manager, gemJoin, end, cdp\)](#) [X](#), 544
[DssProxyActions.freeGem\(manager, gemJoin, cdp, amt\)](#) [X](#), 535

[frob](#)
[call](#)
[D3MHub._exec\(ilk, _pool, Art, lineWad\)](#), 322
[DssCdpManager.frob\(cdp, dink, dart\)](#) [X](#), 524
[DssProxyActions.draw\(manager, jug, daiJoin, cdp, wad\)](#) [X](#), 536
[DssProxyActions.freeETH\(manager, ethJoin, cdp, wad\)](#) [X](#), 535
[DssProxyActions.freeGem\(manager, gemJoin, cdp, amt\)](#) [X](#), 535
[DssProxyActions.frob\(manager, cdp, dink, dart\)](#) [X](#), 533
[DssProxyActions.lockETH\(manager, ethJoin, cdp\)](#) [X](#), 534
[DssProxyActions.lockETHAndDraw\(manager, jug, ethJoin, daiJoin, cdp, wadD\)](#) [X](#), 538
[DssProxyActions.lockGem\(manager, gemJoin, cdp, amt, transferFrom\)](#) [X](#), 535
[DssProxyActions.lockGemAndDraw\(manager, jug, gemJoin, daiJoin, cdp, amtC, wadD, transferFrom\)](#) [X](#), 539
[DssProxyActions.wipe\(manager, daiJoin, cdp, wad\)](#) [X](#), 537
[DssProxyActions.wipeAll\(manager, daiJoin, cdp\)](#) [X](#), 537
[DssProxyActions.wipeAllAndFreeETH\(manager, ethJoin, daiJoin, cdp, wadC\)](#) [X](#), 540
[DssProxyActions.wipeAllAndFreeGem\(manager, gemJoin, daiJoin, cdp, amtC\)](#) [X](#), 541
[DssProxyActions.wipeAndFreeETH\(manager, ethJoin, daiJoin, cdp, wadC, wadD\)](#) [X](#), 540
[DssProxyActions.wipeAndFreeGem\(manager, gemJoin, daiJoin, cdp, amtC, wadD\)](#) [X](#), 541
[DssPsm.buyGem\(usr, gemAmt\)](#) [X](#), 511
[DssPsm.sellGem\(usr, gemAmt\)](#) [X](#), 510
[TeleportJoin._mint\(teleportGUID, hashGUID, maxFeePercentage, operatorFee\)](#), 643
[TeleportJoin.settle\(sourceDomain, batchedDaiToFlush\)](#) [X](#), 645
[definition](#)
[DssCdpManager.frob\(cdp, dink, dart\)](#) [X](#), 524
[DssProxyActions.frob\(manager, cdp, dink, dart\)](#) [X](#), 533
[Vat.frob\(i, u, v, w, dink, dart\)](#) [X](#), 232

[_fromWad](#)
[call](#)
[CalleeMakerOtcDai.clipperCall\(sender, daiAmt, gemAmt, data\)](#) [X](#), 550
[CurveLpTokenUniv3Callee.clipperCall\(sender, owe, slice, data\)](#) [X](#), 552
[PSMCallee.clipperCall\(sender, owe, slice, data\)](#) [X](#), 555
[UniswapV2CalleeDai.clipperCall\(sender, daiAmt, gemAmt, data\)](#) [X](#), 558
[UniswapV2LpTokenCalleeDai.clipperCall\(sender, daiAmt, gemAmt, data\)](#) [X](#), 560
[UniswapV3Callee.clipperCall\(sender, owe, slice, data\)](#) [X](#), 567

WstETHCurveUniv3Callee.clipperCall(**sender**, owe, slice, **data**) **X**, 564

definition

CalleeMakerOtcDai.fromWad(gemJoin, wad) [CalleeMakerOtc], 550

CurveLpTokenUniv3Callee.fromWad(gemJoin, wad), 552

PSMCallee.fromWad(gemJoin, wad), 555

UniswapV2CalleeDai.fromWad(gemJoin, wad) [UniswapV2Callee], 557

UniswapV2LpTokenCalleeDai.fromWad(gemJoin, wad) [UniswapV2Callee(2)], 559

UniswapV3Callee.fromWad(gemJoin, wad), 567

WstETHCurveUniv3Callee.fromWad(gemJoin, wad), 563

functionCall

call

Address.functionCall(target, **data**), 764

definition

Address.functionCall(target, **data**), 764

Address.functionCall(target, **data**, errorMessage), 764

functionCallWithValue

call

Address.functionCall(target, **data**, errorMessage), 764

Address.functionCallWithValue(target, **data**, **value**), 764

definition

Address.functionCallWithValue(target, **data**, **value**), 764

Address.functionCallWithValue(target, **data**, **value**, errorMessage), 765

functionDelegateCall

call

Address.functionDelegateCall(target, **data**), 765

definition

Address.functionDelegateCall(target, **data**), 765

Address.functionDelegateCall(target, **data**, errorMessage), 766

functionStaticCall

call

Address.functionStaticCall(target, **data**), 765

definition

Address.functionStaticCall(target, **data**), 765

Address.functionStaticCall(target, **data**, errorMessage), 765

G

gap

write

End.skim(ilk, urn) **X**, 300

definition

End.gap: **mapping**(bytes32 => uint256), 295

read

DssAutoLine.exec(_ilk) **X**, 707

End.flow(ilk) **X**, 300

End.skim(ilk, urn) **X**, 300

gasMargin

write

TrustedRelay.file(what, **data**) **X** **a**, 662

definition

TrustedRelay.gasMargin: **uint256**, 661

read

TrustedRelay.relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s) **X**, 663

gasMeteringOff

write

D3MCoreDeployScript.**modifier** noGasMetering() [StdCheatsSafe], 335

D3MCoreInitScript.**modifier** noGasMetering() [StdCheatsSafe], 363

D3MDeployScript.**modifier** noGasMetering() [StdCheatsSafe], 389

D3MInitScript.**modifier** noGasMetering() [StdCheatsSafe], 409

definition

D3MCoreDeployScript.gasMeteringOff [StdCheatsSafe] : **bool**, 330

D3MCoreInitScript.gasMeteringOff [StdCheatsSafe] : **bool**, 358

D3MDeployScript.gasMeteringOff [StdCheatsSafe] : **bool**, 384

D3MInitScript.gasMeteringOff [StdCheatsSafe] : **bool**, 404

read

D3MCoreDeployScript.**modifier** noGasMetering() [StdCheatsSafe], 335

D3MCoreInitScript.**modifier** noGasMetering() [StdCheatsSafe], 363

D3MDeployScript.**modifier** noGasMetering() [StdCheatsSafe], 389

D3MInitScript.**modifier** noGasMetering() [StdCheatsSafe], 409

gasprice

call

TrustedRelay.relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s) **X**, 663

definition

TrustedRelay.**gasprice**(), 664

gateways

write

TeleportRouter.file(what, domain, **data**) **X** **a**, 653

definition

TeleportRouter.gateways: **mapping**(bytes32 => **address**), 653

read

TeleportRouter.file(what, domain, **data**) **X** **a**, 653

TeleportRouter.requestMint(teleportGUID, maxFeePercentage, operatorFee) **X**, 654

TeleportRouter.settle(targetDomain, batchedDaiToFlush) **X**, 655

gem

call

CalleeMakerOtcDai.clipperCall(**sender**, daiAmt, gemAmt, **data**) **X**, 550

CurveLpTokenUniv3Callee.clipperCall(**sender**, owe, slice, **data**) **X**, 552

D3MHub.uncull(ilk) **X**, 326

D3MHub.wipe(ilk, _pool, urn), 322

DssExecLib.addCollateralBase(_ilk, _gem, _join, _clip, _calc, _pip) **X**, 749

DssProxyActionsEnd.cashETH(ethJoin, end, ilk, wad) **X**, 544

DssProxyActionsEnd.freeETH(manager, ethJoin, end, cdp) **X**, 544

DssProxyActions.ethJoin_join(apt, urn) **X**, 531

DssProxyActions.exitETH(manager, ethJoin, cdp, wad) **X**, 536

DssProxyActions.freeETH(manager, ethJoin, cdp, wad) **X**, 535

DssProxyActions.gemJoin_join(apt, urn, amt, transferFrom) **X**, 531

DssProxyActions.openLockGNTAndDraw(manager, jug, gntJoin, daiJoin, ilk, amtC, wadD) [X](#), 539
 DssProxyActions.wipeAllAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC) [X](#), 540
 DssProxyActions.wipeAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC, wadD) [X](#), 540
 IlkRegistry.add(adapter) [X](#), 668
 PSMCallee.clipperCall(sender, owe, slice, data) [X](#), 555
 UniswapV2CalleeDai.clipperCall(sender, daiAmt, gemAmt, data) [X](#), 558
 UniswapV2LpTokenCalleeDai.clipperCall(sender, daiAmt, gemAmt, data) [X](#), 560
 UniswapV3Callee.clipperCall(sender, owe, slice, data) [X](#), 567
 WstETHCurveUniv3Callee.clipperCall(sender, owe, slice, data) [X](#), 564

write

AuthGemJoin5.constructor(vat_, ilk_, gem_) [X](#), 503
 AuthGemJoin8.constructor(vat_, ilk_, gem_) [X](#), 505
 AuthGemJoin.constructor(vat_, ilk_, gem_) [X](#), 447, 507
 DssVestMintable.constructor(_gem) [X](#), 683
 DssVestTransferrable.constructor(_czar, _gem) [X](#), 703
 ESM.constructor(gem_, end_, proxy_, min_) [X](#), 303
 Flapper.constructor(vat_, gem_) [X](#), 262
 Flopper.constructor(vat_, gem_) [X](#), 266
 GemBag.constructor(lad_, gem_) [X](#), 431
 GemJoin2.constructor(vat_, ilk_, gem_) [X](#), 427
 GemJoin3.constructor(vat_, ilk_, gem_, decimals) [X](#), 429
 GemJoin4.constructor(vat_, ilk_, gem_) [X](#), 433
 GemJoin5.constructor(vat_, ilk_, gem_) [X](#), 435
 GemJoin6.constructor(vat_, ilk_, gem_) [X](#), 437
 GemJoin7.constructor(vat_, ilk_, gem_) [X](#), 439
 GemJoin8.constructor(vat_, ilk_, gem_) [X](#), 441
 GemJoin9.constructor(vat_, ilk_, gem_) [X](#), 443
 GemJoin.constructor(vat_, ilk_, gem_) [X](#), 424
 IlkRegistry.file(ilk, what, data) [X](#) [a](#), 669
 ManagedGemJoin.constructor(vat_, ilk_, gem_) [X](#), 449
 Vat.flux(ilk, src, dst, wad) [X](#), 231
 Vat.frob(i, u, v, w, dink, dart) [X](#), 232
 Vat.grab(i, u, v, w, dink, dart) [X](#) [a](#), 233
 Vat.slip(ilk, usr, wad) [X](#) [a](#), 231

definition

AuthGemJoin5.gem: GemLike, 503
 AuthGemJoin8.gem: GemLike, 505
 AuthGemJoin.gem: GemLike, 447, 507
 DssVestMintable.gem: MintLike, 675
 DssVestTransferrable.gem: TokenLike, 695
 ESM.gem: GemLike, 303
 Flapper.gem: GemLike, 261
 Flopper.gem: GemLike, 265
 GemBag.gem: GemLike, 431
 GemJoin2.gem: GemLike, 427
 GemJoin3.gem: GemLike, 429
 GemJoin4.gem: GemLike, 433
 GemJoin5.gem: GemLike, 435
 GemJoin6.gem: GemLike, 437
 GemJoin7.gem: GemLike, 439
 GemJoin8.gem: GemLike, 441
 GemJoin9.gem: GemLike, 443
 GemJoin.gem: GemLike, 423
 IlkRegistry.gem(ilk), 672
 ManagedGemJoin.gem: GemLike, 449
 Vat.gem: [mapping](#)(bytes32 => [mapping](#)(address => uint256)), 229

read

AuthGemJoin5.exit(usr, amt) [X](#), 504
 AuthGemJoin5.join(urn, amt, msgSender) [X](#) [a](#), 504
 AuthGemJoin8.exit(usr, amt) [X](#), 506
 AuthGemJoin8.join(urn, amt, msgSender) [X](#) [a](#), 506
 AuthGemJoin.constructor(vat_, ilk_, gem_) [X](#), 447

AuthGemJoin.exit(guy, wad) [X](#), 508
 AuthGemJoin.exit(usr, wad) [X](#), 448
 AuthGemJoin.join(urn, wad, msgSender) [X a](#), 508
 AuthGemJoin.join(usr, wad) [X a](#), 448
 DssVestMintable.pay(_guy, _amt), 683
 DssVestTransferrable.pay(_guy, _amt), 703
 ESM.burn() [X](#), 305
 ESM.join(wad) [X](#), 305
 Flapper.deal(id) [X](#), 264
 Flapper.tend(id, lot, bid) [X](#), 263
 Flapper.yank(id) [X](#), 264
 Flopper.deal(id) [X](#), 267
 GemBag.exit(usr, wad) [X](#), 431
 GemJoin2.constructor(vat_, ilk_, gem_) [X](#), 427
 GemJoin2.exit(usr, wad) [X](#), 428
 GemJoin2.join(usr, wad) [X](#), 428
 GemJoin3.exit(usr, amt) [X](#), 430
 GemJoin3.join(usr, amt) [X](#), 430
 GemJoin4.constructor(vat_, ilk_, gem_) [X](#), 433
 GemJoin4.exit(usr, wad) [X](#), 434
 GemJoin4.make(usr) [X](#), 434
 GemJoin5.constructor(vat_, ilk_, gem_) [X](#), 435
 GemJoin5.exit(usr, amt) [X](#), 436
 GemJoin5.join(usr, amt) [X](#), 436
 GemJoin6.constructor(vat_, ilk_, gem_) [X](#), 437
 GemJoin6.exit(usr, wad) [X](#), 438
 GemJoin6.join(usr, wad) [X](#), 438
 GemJoin7.constructor(vat_, ilk_, gem_) [X](#), 439
 GemJoin7.exit(usr, amt) [X](#), 440
 GemJoin7.join(usr, amt) [X](#), 440
 GemJoin8.constructor(vat_, ilk_, gem_) [X](#), 441
 GemJoin8.exit(usr, amt) [X](#), 442
 GemJoin8.join(usr, amt) [X](#), 442
 GemJoin9.exit(usr, wad) [X](#), 445
 GemJoin9._join(usr), 444
 GemJoin9.join(usr, wad) [X](#), 444
 GemJoin.constructor(vat_, ilk_, gem_) [X](#), 424
 GemJoin.exit(usr, wad) [X](#), 424
 GemJoin.join(usr, wad) [X](#), 424
 IlkRegistry.gem(ilk), 672
 ManagedGemJoin.exit(urn, usr, amt) [X a](#), 450
 ManagedGemJoin.join(usr, amt) [X a](#), 450
 Vat.flux(ilk, src, dst, wad) [X](#), 231
 Vat.frob(i, u, v, w, dink, dart) [X](#), 232
 Vat.grab(i, u, v, w, dink, dart) [X a](#), 233
 Vat.slip(ilk, usr, wad) [X a](#), 231

GemBag
 [definition](#)
 [contract](#) GemBag, 431

GemInfo
 [definition](#)
 [contract](#) GemInfo, 665

gemInfo
 [write](#)
 IlkRegistry.constructor(vat_, dog_, cat_, spot_) [X](#), 668
 [definition](#)
 IlkRegistry.gemInfo: GemInfo, 667
 [read](#)
 IlkRegistry.add(adapter) [X](#), 668

GemJoin
 [definition](#)
 [contract](#) GemJoin, 423

gemJoin

call
 PSMCallee.clipperCall(sender, owe, slice, data) X, 555
 write
 DssPsm.constructor(gemJoin_, daiJoin_, vow_) X, 509
 definition
 DssPsm.gemJoin: AuthGemJoinAbstract, 509
 read
 DssPsm.constructor(gemJoin_, daiJoin_, vow_) X, 509
 DssPsm.buyGem(usr, gemAmt) X, 511
 DssPsm.sellGem(usr, gemAmt) X, 510
 GemJoin2
 definition
 contract GemJoin2, 427
 GemJoin3
 definition
 contract GemJoin3, 429
 GemJoin4
 definition
 contract GemJoin4, 433
 GemJoin5
 definition
 contract GemJoin5, 435
 GemJoin6
 definition
 contract GemJoin6, 437
 GemJoin7
 definition
 contract GemJoin7, 439
 GemJoin8
 definition
 contract GemJoin8, 441
 GemJoin9
 definition
 contract GemJoin9, 443
 gemJoin_join
 call
 DssProxyActions.lockGem(manager, gemJoin, cdp, amt, transferFrom) X, 535
 DssProxyActions.lockGemAndDraw(manager, jug, gemJoin, daiJoin, cdp, amtC, wadD,
 transferFrom) X, 539
 definition
 DssProxyActions.gemJoin_join(apt, urn, amt, transferFrom) X, 531
 get
 definition
 ChainLog.get(_index), 711
 IlkRegistry.get(pos), 671
 getAccountSnapshot
 call
 D3MCompoundPool.assetBalance(), 379
 getAddress
 call
 D3MDeployScript.run() X, 400
 D3MInit._init(dss, d3m, cfg, gem), 725
 D3MInit.initCore(dss, d3mCore), 725
 DssExec.constructor(_expiration, _spellAction) X, 309
 DssExecLib.getChangelogAddress(_key), 733
 DssVestSuckable.constructor(_chainlog) X, 692
 DssVestSuckable.pay(_guy, _amt), 693
 MCD.getAddressOrNull(dss, key), 719
 definition
 ChainLog.getAddress(_key), 711
 getAddressOrNull
 call
 MCD.getIlk(dss, gem, variant), 720

- MCD.loadFromChainlog(chainlog), 719
 - definition
- MCD.getAddressOrNull(dss, key), 719
- getCash
 - call
 - D3MCompoundPlan.getTargetAssets(currentAssets), 317
 - D3MCompoundPool.maxWithdraw(), 379
- GetCdps
 - definition
 - contract GetCdps, 527
- getCdpsAsc
 - definition
 - GetCdps.getCdpsAsc(manager, guy), 527
- getCdpsDesc
 - definition
 - GetCdps.getCdpsDesc(manager, guy), 527
- getChain
 - definition
 - D3MCoreDeployScript.getChain(chainAlias) [StdChains], 343
 - D3MCoreDeployScript.getChain(chainId) [StdChains], 343
 - D3MCoreInitScript.getChain(chainAlias) [StdChains], 371
 - D3MCoreInitScript.getChain(chainId) [StdChains], 371
 - D3MDeployScript.getChain(chainAlias) [StdChains], 397
 - D3MDeployScript.getChain(chainId) [StdChains], 397
 - D3MInitScript.getChain(chainAlias) [StdChains], 417
 - D3MInitScript.getChain(chainId) [StdChains], 418
- getChainWithUpdatedRpcUrl
 - call
 - D3MCoreDeployScript.getChain(chainAlias) [StdChains], 343
 - D3MCoreDeployScript.getChain(chainId) [StdChains], 343
 - D3MCoreInitScript.getChain(chainAlias) [StdChains], 371
 - D3MCoreInitScript.getChain(chainId) [StdChains], 371
 - D3MDeployScript.getChain(chainAlias) [StdChains], 397
 - D3MDeployScript.getChain(chainId) [StdChains], 397
 - D3MInitScript.getChain(chainAlias) [StdChains], 417
 - D3MInitScript.getChain(chainId) [StdChains], 418
 - definition
 - D3MCoreDeployScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 345
 - D3MCoreInitScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 373
 - D3MDeployScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 399
 - D3MInitScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 419
- getChangelogAddress
 - call
 - DssExecLib.autoLine(), 732
 - DssExecLib.cat(), 730
 - DssExecLib.clipperMom(), 732
 - DssExecLib.dai(), 730
 - DssExecLib.daiJoin(), 732
 - DssExecLib.dog(), 730
 - DssExecLib.end(), 731
 - DssExecLib.esm(), 731
 - DssExecLib.flap(), 731
 - DssExecLib.flipperMom(), 732
 - DssExecLib.flop(), 731
 - DssExecLib.govGuard(), 732
 - DssExecLib.jug(), 731
 - DssExecLib.lerpFab(), 732
 - DssExecLib.mkr(), 730
 - DssExecLib.osmMom(), 731
 - DssExecLib.pauseProxy(), 732
 - DssExecLib.pot(), 731
 - DssExecLib.reg(), 731
 - DssExecLib.setRWAILkDebtCeiling(_ilk, _ceiling, _price) X, 742

DssExecLib.spotter(), 731

DssExecLib.vat(), 730

DssExecLib.vow(), 731

definition

DssExecLib.getChangelogAddress(_key), 733

getCode

call

D3MCoreDeployScript.deployCode(what) [StdCheatsSafe], 342

D3MCoreDeployScript.deployCode(what, args) [StdCheatsSafe], 341

D3MCoreDeployScript.deployCode(what, args, val) [StdCheatsSafe], 342

D3MCoreDeployScript.deployCode(what, val) [StdCheatsSafe], 342

D3MCoreInitScript.deployCode(what) [StdCheatsSafe], 370

D3MCoreInitScript.deployCode(what, args) [StdCheatsSafe], 369

D3MCoreInitScript.deployCode(what, args, val) [StdCheatsSafe], 370

D3MCoreInitScript.deployCode(what, val) [StdCheatsSafe], 370

D3MDeployScript.deployCode(what) [StdCheatsSafe], 396

D3MDeployScript.deployCode(what, args) [StdCheatsSafe], 395

D3MDeployScript.deployCode(what, args, val) [StdCheatsSafe], 396

D3MDeployScript.deployCode(what, val) [StdCheatsSafe], 396

D3MInitScript.deployCode(what) [StdCheatsSafe], 416

D3MInitScript.deployCode(what, args) [StdCheatsSafe], 416

D3MInitScript.deployCode(what, args, val) [StdCheatsSafe], 416

D3MInitScript.deployCode(what, val) [StdCheatsSafe], 416

getCompAddress

call

D3MCompoundPool.constructor(ilk_, hub_, cDai_) X, 377

getCrossDomainMessenger

call

L1DAITokenBridge.modifier onlyFromCrossDomainAccount(_sourceDomainAccount)

[OVM_CrossDomainEnabled], 592

L1DAITokenBridge.sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message)

[OVM_CrossDomainEnabled], 593

L1GovernanceRelay.modifier onlyFromCrossDomainAccount(_sourceDomainAccount)

[OVM_CrossDomainEnabled], 599

L1GovernanceRelay.sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message)

[OVM_CrossDomainEnabled], 600

L2DAITokenBridge.modifier onlyFromCrossDomainAccount(_sourceDomainAccount)

[OVM_CrossDomainEnabled], 603

L2DAITokenBridge.sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message)

[OVM_CrossDomainEnabled], 604

L2GovernanceRelay.modifier onlyFromCrossDomainAccount(_sourceDomainAccount)

[OVM_CrossDomainEnabled], 609

L2GovernanceRelay.sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message)

[OVM_CrossDomainEnabled], 610

definition

L1DAITokenBridge.getCrossDomainMessenger() [OVM_CrossDomainEnabled], 592

L1GovernanceRelay.getCrossDomainMessenger() [OVM_CrossDomainEnabled], 600

L2DAITokenBridge.getCrossDomainMessenger() [OVM_CrossDomainEnabled], 604

L2GovernanceRelay.getCrossDomainMessenger() [OVM_CrossDomainEnabled], 610

getDrawDart

call

DssProxyActions.draw(manager, jug, daiJoin, cdp, wad) X, 536

DssProxyActions.lockETHAndDraw(manager, jug, ethJoin, daiJoin, cdp, wadD) X, 538

DssProxyActions.lockGemAndDraw(manager, jug, gemJoin, daiJoin, cdp, amtC, wadD, transferFrom) X, 539

definition

DssProxyActions._getDrawDart(vat, jug, urn, ilk, wad), 530

getFee

call

TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643

definition

TeleportConstantFee.getFee(guid, , , , amtToTake), 639

TeleportLinearFee.getFee(guid, , , , amtToTake), 647

getFeedPrice

call

Clipper.kick(tab, lot, usr, kpr) [X](#), [a](#), 244

Clipper.redo(id, kpr) [X](#), 245

definition

Clipper.getFeedPrice(), 244

getFeeFor

call

PAXG.transferFrom(src, dst, wad) [X](#), 478

definition

PAXG.getFeeFor(_value), 479

getGUIDHash

call

BasicRelay.relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s) [X](#), 658

TeleportJoin.mintPending(teleportGUID, maxFeePercentage, operatorFee) [X](#), 644

TeleportJoin.requestMint(teleportGUID, maxFeePercentage, operatorFee) [X](#), [a](#), 644

TeleportOracleAuth.getSignHash(teleportGUID), 651

TrustedRelay.requestMint(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s), 664

getIlk

definition

MCD.getIlk(dss, gem, variant), 720

getIncentivesController

call

D3MAavePool.collect() [X](#), 352

getMaxVariableBorrowRate

call

D3MAavePlan._calculateTargetSupply(targetInterestRate, totalDebt), 313

getOutboundCalldata

call

L1DaiGateway.outboundTransfer(l1Token, to, amount, maxGas, gasPriceBid, [data](#)) [X](#), 571

L2DaiGateway.outboundTransfer(l1Token, to, amount, , , [data](#)) [X](#), 583

definition

L1DaiGateway.getOutboundCalldata(l1Token, from, to, amount, [data](#)), 572

L2DaiGateway.getOutboundCalldata(token, from, to, amount, [data](#)), 583

getPrices

call

ClipperMom.tripBreaker(clip) [X](#), 259

definition

ClipperMom.getPrices(clip), 258

getReserveData

call

D3MAavePlan.active(), 314

D3MAavePlan.constructor(dai_, pool_) [X](#), 311

D3MAavePool.constructor(ilk_, hub_, dai_, pool_) [X](#), 349

getReserveNormalizedIncome

call

D3MAavePool.deposit(wad) [X](#), 350

getRootChainId

call

ScriptTools.exportContract(name, label, addr), 715

ScriptTools.readInput(name), 713

ScriptTools.readOutput(name), 714

ScriptTools.readOutput(name, [timestamp](#)), 713

definition

ScriptTools.getRootChainId(), 713

getSignHash

call

TeleportOracleAuth.requestMint(teleportGUID, signatures, maxFeePercentage, operatorFee) [X](#), 650

definition

TeleportOracleAuth.getSignHash(teleportGUID), 651
 getStatus
 definition
 Clipper.getStatus(id), 248
 getTargetAssets
 call
 D3MHub._exec(ilk, _pool, Art, lineWad), 322
 definition
 D3MAavePlan.getTargetAssets(currentAssets), 313
 D3MCompoundPlan.getTargetAssets(currentAssets), 317
 _getWipeAllWad
 call
 DssProxyActions.wipeAll(manager, daiJoin, cdp) **X**, 537
 DssProxyActions.wipeAllAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC) **X**, 540
 DssProxyActions.wipeAllAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC) **X**, 541
 definition
 DssProxyActions._getWipeAllWad(vat, usr, urn, ilk), 530
 _getWipeDart
 call
 DssProxyActions.wipe(manager, daiJoin, cdp, wad) **X**, 537
 DssProxyActions.wipeAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC, wadD) **X**, 540
 DssProxyActions.wipeAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC, wadD) **X**, 541
 definition
 DssProxyActions._getWipeDart(vat, dai, urn, ilk), 530
 give
 call
 DssProxyActions.give(manager, cdp, usr) **X**, 532
 DssProxyActions.giveToProxy(proxyRegistry, manager, cdp, dst) **X**, 532
 definition
 DssCdpManager.give(cdp, dst) **X**, 523
 DssProxyActions.give(manager, cdp, usr) **X**, 532
 giveAdminAccess
 call
 MCD.giveAdminAccess(dss), 721
 definition
 MCD.giveAdminAccess(dss), 721
 MCD.giveAdminAccess(dss, who), 721
 giveToProxy
 definition
 DssProxyActions.giveToProxy(proxyRegistry, manager, cdp, dst) **X**, 532
 GNT
 definition
 contract GNT, 461
 GodMode
 definition
 library GodMode, 767
 govGuard
 definition
 DssExecLib.govGuard(), 732
 grab
 call
 D3MHub.cull(ilk) **X**, 326
 D3MHub._exec(ilk, _pool, Art, lineWad), 322
 D3MHub.uncull(ilk) **X**, 326
 Dog.bark(ilk, urn, kpr) **X**, 253
 End.free(ilk) **X**, 300
 End.skim(ilk, urn) **X**, 300
 End.skip(ilk, id) **X**, 299
 End.snip(ilk, id) **X**, 299
 definition
 Vat.grab(i, u, v, w, dink, dart) **X** **a**, 233
 GUSD
 definition

```
    contract GUSD, 463
guy
  write
    Flapper.kick(lot, bid) X a, 263
    Flapper.tend(id, lot, bid) X, 263
    Flopper.dent(id, lot, bid) X, 267
    Flopper.kick(gal, lot, bid) X a, 266
  read
    Flapper.deal(id) X, 264
    Flapper.tend(id, lot, bid) X, 263
    Flapper.yank(id) X, 264
    Flopper.deal(id) X, 267
    Flopper.dent(id, lot, bid) X, 267
    Flopper.yank(id) X, 267
```

H**HANDLE_FLUSH**

definition

L1DAITeleportGateway.HANDLE_FLUSH: [uint256](#), 623

read

L1DAITeleportGateway.finalizeFlush(targetDomain, daiToFlush) [X](#), 623

HANDLE_REGISTER_TELEPORT

definition

L1DAITeleportGateway.HANDLE_REGISTER_TELEPORT: [uint256](#), 623

read

L1DAITeleportGateway.finalizeRegisterTeleport(teleport) [X](#), 623

HANDLE_WITHDRAW

definition

L1DAIBridge.HANDLE_WITHDRAW: [uint256](#), 617

read

L1DAIBridge.withdraw(amount, l1Recipient) [X](#), 619

has

definition

Value.has: [bool](#), 289

read

Value.peek(), 289

hasDomain

definition

TeleportRouter.hasDomain(domain), 654

heal

call

Vow.cage() [X](#) [a](#), 271

Vow.heal(rad) [X](#), 270

Vow.kiss(rad) [X](#), 271

definition

Vat.heal(rad) [X](#), 233

Vow.heal(rad) [X](#), 270

Hole

write

Dog.file(what, [data](#)) [X](#) [a](#), 252

definition

Dog.Hole: [uint256](#), 251

read

Dog.bark(ilk, urn, kpr) [X](#), 253

hole

write

Dog.file(ilk, what, [data](#)) [X](#) [a](#), 253

hop

write

OSM.step(ts) [X](#) [a](#), 287

definition

OSM.hop: [uint16](#), 285

read

OSM.pass(), 287

OSM.prev(ts), 287

hope

call

D3MAavePool.constructor(ilk_, hub_, dai_, pool_) [X](#), 349

D3MAavePool.file(what, [data](#)) [X](#) [a](#), 350

D3MCompoundPool.constructor(ilk_, hub_, cDai_) [X](#), 377

D3MCompoundPool.file(what, [data](#)) [X](#) [a](#), 378

D3MHub.constructor(daiJoin_) [X](#), 320

DssExecLib.delegateVat(_usr) [X](#), 734

DssProxyActions.draw(manager, jug, daiJoin, cdp, wad) [X](#), 536

DssProxyActionsDsr.exit(daiJoin, pot, wad) [X](#), 548

DssProxyActionsDsr.exitAll(daiJoin, pot) [X](#), 548

DssProxyActionsDsr.join(daiJoin, pot, wad) [X](#), 547

DssProxyActionsEnd.free(manager, end, cdp), 543

DssProxyActionsEnd.pack(daiJoin, end, wad) [X](#), 544
 DssProxyActions.hope(obj, usr) [X](#), 531
 DssProxyActions.lockETHAndDraw(manager, jug, ethJoin, daiJoin, cdp, wadD) [X](#), 538
 DssProxyActions.lockGemAndDraw(manager, jug, gemJoin, daiJoin, cdp, amtC, wadD, transferFrom) [X](#), 539
 DssPsm.constructor(gemJoin_, daiJoin_, vow_) [X](#), 509
 DssPsm.hope(usr) [X](#) [a](#), 510
 DssVestSuckable.constructor(_chainlog) [X](#), 692
 End.skip(ilk, id) [X](#), 299
 TeleportJoin.constructor(vat_, daiJoin_, ilk_, domain_) [X](#), 641
 UrnHandler.constructor(vat) [X](#), 519
 Vow.constructor(vat_, flapper_, flopper_) [X](#), 269
 Vow.file(what, [data](#)) [X](#) [a](#), 270

definition

DssProxyActions.hope(obj, usr) [X](#), 531
 DssPsm.hope(usr) [X](#) [a](#), 510
 Vat.hope(usr) [X](#), 230

hub

call

D3MInit.initAave(dss, d3m, cfg, aaveCfg), 726
 D3MInit.initCompound(dss, d3m, cfg, compoundCfg), 727

write

D3MAavePool.constructor(ilk_, hub_, dai_, pool_) [X](#), 349
 D3MAavePool.file(what, [data](#)) [X](#) [a](#), 350
 D3MCompoundPool.constructor(ilk_, hub_, cDai_) [X](#), 377
 D3MCompoundPool.file(what, [data](#)) [X](#) [a](#), 378
 D3MDeployScript.run() [X](#), 400
 D3MOracle.file(what, [data](#)) [X](#) [a](#), 354

definition

D3MAavePool.hub: [address](#), 349
 D3MCompoundPool.hub: [address](#), 377
 D3MDeployScript.hub: [address](#), 383
 D3MOracle.hub: [address](#), 353

read

D3MAavePool.modifier onlyHub(), 349
 D3MAavePool.exit(dst, wad) [X](#), 351
 D3MAavePool.file(what, [data](#)) [X](#) [a](#), 350
 D3MCompoundPool.modifier onlyHub(), 377
 D3MCompoundPool.exit(dst, wad) [X](#), 379
 D3MCompoundPool.file(what, [data](#)) [X](#) [a](#), 378
 D3MDeployScript.run() [X](#), 400
 D3MOracle.peek(), 354

hump

write

Vow.file(what, [data](#)) [X](#) [a](#), 270

definition

Vow.hump: [uint256](#), 269

read

Vow.flap() [X](#), 271

I

ids

write

DssVestMintable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] **X** **a**, 678
 DssVestSuckable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] **X** **a**, 688
 DssVestTransferrable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] **X** **a**, 698

definition

DssVestMintable.ids [DssVest] : **uint256**, 675
 DssVestSuckable.ids [DssVest] : **uint256**, 685
 DssVestTransferrable.ids [DssVest] : **uint256**, 695

read

DssVestMintable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] **X** **a**, 678
 DssVestSuckable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] **X** **a**, 688
 DssVestTransferrable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] **X** **a**, 698

idToAlias

write

D3MCoreDeployScript.setChain(chainAlias, chain) [StdChains], 344
 D3MCoreInitScript.setChain(chainAlias, chain) [StdChains], 372
 D3MDeployScript.setChain(chainAlias, chain) [StdChains], 398
 D3MInitScript.setChain(chainAlias, chain) [StdChains], 418

definition

D3MCoreDeployScript.idToAlias [StdChains] : **mapping(uint256 => string)**, 330
 D3MCoreInitScript.idToAlias [StdChains] : **mapping(uint256 => string)**, 358
 D3MDeployScript.idToAlias [StdChains] : **mapping(uint256 => string)**, 384
 D3MInitScript.idToAlias [StdChains] : **mapping(uint256 => string)**, 404

read

D3MCoreDeployScript.getChain(chainId) [StdChains], 343
 D3MCoreDeployScript.setChain(chainAlias, chain) [StdChains], 344
 D3MCoreInitScript.getChain(chainId) [StdChains], 371
 D3MCoreInitScript.setChain(chainAlias, chain) [StdChains], 372
 D3MDeployScript.getChain(chainId) [StdChains], 397
 D3MDeployScript.setChain(chainAlias, chain) [StdChains], 398
 D3MInitScript.getChain(chainId) [StdChains], 418
 D3MInitScript.setChain(chainAlias, chain) [StdChains], 418

Ilk

definition

struct D3MHub.Ilks, 319
struct Dog.Ilks, 251
struct DssAutoLine.Ilks, 705
struct IlkRegistry.Ilks, 667
struct Jug.Ilks, 273
struct Spotter.Ilks, 227
struct Vat.Ilks, 229

ilk

call

ClipperMom.getPrices(clip), 258
 D3MInit.init(dss, d3m, cfg, gem), 725
 D3MInit.initAave(dss, d3m, cfg, aaveCfg), 726
 D3MInit.initCompound(dss, d3m, cfg, compoundCfg), 727
 Dog.file(ilk, what, clip) **X** **a**, 253
 DssExecLib.addCollateralBase(_ilk, _gem, _join, _clip, _calc, _pip) **X**, 749
 DssPsm.constructor(gemJoin_, daiJoin_, vow_) **X**, 509
 IlkRegistry.add(adapter) **X**, 668

write

AuthGemJoin5.constructor(vat_, ilk_, gem_) **X**, 503
 AuthGemJoin8.constructor(vat_, ilk_, gem_) **X**, 505
 AuthGemJoin.constructor(vat_, ilk_, gem_) **X**, 447, 507
 Clipper.constructor(vat_, spotter_, dog_, ilk_) **X**, 243
 D3MAavePool.constructor(ilk_, hub_, dai_, pool_) **X**, 349
 D3MCompoundPool.constructor(ilk_, hub_, cDai_) **X**, 377
 D3MDeployScript.run() **X**, 400
 D3MInitScript.run() **X**, 420
 D3MOracle.constructor(vat_, ilk_) **X**, 353

DssPsm.[constructor](#)(gemJoin_, daiJoin_, vow_) [X](#), 509
 GemJoin2.[constructor](#)(vat_, ilk_, gem_) [X](#), 427
 GemJoin3.[constructor](#)(vat_, ilk_, gem_, decimals) [X](#), 429
 GemJoin4.[constructor](#)(vat_, ilk_, gem_) [X](#), 433
 GemJoin5.[constructor](#)(vat_, ilk_, gem_) [X](#), 435
 GemJoin6.[constructor](#)(vat_, ilk_, gem_) [X](#), 437
 GemJoin7.[constructor](#)(vat_, ilk_, gem_) [X](#), 439
 GemJoin8.[constructor](#)(vat_, ilk_, gem_) [X](#), 441
 GemJoin9.[constructor](#)(vat_, ilk_, gem_) [X](#), 443
 GemJoin.[constructor](#)(vat_, ilk_, gem_) [X](#), 424
 ManagedGemJoin.[constructor](#)(vat_, ilk_, gem_) [X](#), 449
 TeleportJoin.[constructor](#)(vat_, daiJoin_, ilk_, domain_) [X](#), 641

definition

AuthGemJoin5.ilk: [bytes32](#), 503
 AuthGemJoin8.ilk: [bytes32](#), 505
 AuthGemJoin.ilk: [bytes32](#), 447, 507
 Clipper.ilk: [bytes32](#), 241
 D3MAavePool.ilk: [bytes32](#), 349
 D3MCompoundPool.ilk: [bytes32](#), 377
 D3MDeployScript.ilk: [bytes32](#), 383
 D3MInitScript.ilk: [bytes32](#), 403
 D3MOracle.ilk: [bytes32](#), 353
 DssPsm.ilk: [bytes32](#), 509
 GemJoin2.ilk: [bytes32](#), 427
 GemJoin3.ilk: [bytes32](#), 429
 GemJoin4.ilk: [bytes32](#), 433
 GemJoin5.ilk: [bytes32](#), 435
 GemJoin6.ilk: [bytes32](#), 437
 GemJoin7.ilk: [bytes32](#), 439
 GemJoin8.ilk: [bytes32](#), 441
 GemJoin9.ilk: [bytes32](#), 443
 GemJoin.ilk: [bytes32](#), 423
 ManagedGemJoin.ilk: [bytes32](#), 449
 TeleportJoin.ilk: [bytes32](#), 641

read

AuthGemJoin5.exit(usr, amt) [X](#), 504
 AuthGemJoin5.join(urn, amt, msgSender) [X](#) [a](#), 504
 AuthGemJoin8.exit(usr, amt) [X](#), 506
 AuthGemJoin8.join(urn, amt, msgSender) [X](#) [a](#), 506
 AuthGemJoin.exit(guy, wad) [X](#), 508
 AuthGemJoin.exit(usr, wad) [X](#), 448
 AuthGemJoin.join(urn, wad, msgSender) [X](#) [a](#), 508
 AuthGemJoin.join(usr, wad) [X](#) [a](#), 448
 Clipper.getFeedPrice(), 244
 Clipper.take(id, amt, max, who, [data](#)) [X](#), 246
 Clipper.upchost() [X](#), 248
 Clipper.yank(id) [X](#) [a](#), 249
 D3MAavePool.exit(dst, wad) [X](#), 351
 D3MCompoundPool.exit(dst, wad) [X](#), 379
 D3MDeployScript.run() [X](#), 400
 D3MInitScript.run() [X](#), 420
 D3MOracle.peek(), 354
 DssPsm.buyGem(usr, gemAmt) [X](#), 511
 DssPsm.sellGem(usr, gemAmt) [X](#), 510
 GemJoin2.exit(usr, wad) [X](#), 428
 GemJoin2.join(usr, wad) [X](#), 428
 GemJoin3.exit(usr, amt) [X](#), 430
 GemJoin3.join(usr, amt) [X](#), 430
 GemJoin4.exit(usr, wad) [X](#), 434
 GemJoin4.join(usr, wad) [X](#), 434
 GemJoin5.exit(usr, amt) [X](#), 436
 GemJoin5.join(usr, amt) [X](#), 436
 GemJoin6.exit(usr, wad) [X](#), 438

GemJoin6.join(usr, wad) [X](#), 438
 GemJoin7.exit(usr, amt) [X](#), 440
 GemJoin7.join(usr, amt) [X](#), 440
 GemJoin8.exit(usr, amt) [X](#), 442
 GemJoin8.join(usr, amt) [X](#), 442
 GemJoin9.exit(usr, wad) [X](#), 445
 GemJoin9._join(usr), 444
 GemJoin.exit(usr, wad) [X](#), 424
 GemJoin.join(usr, wad) [X](#), 424
 ManagedGemJoin.exit(urn, usr, amt) [X](#) [a](#), 450
 ManagedGemJoin.join(usr, amt) [X](#) [a](#), 450
 TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643
 TeleportJoin.settle(sourceDomain, batchedDaiToFlush) [X](#), 645

ilkData

write

IlkRegistry.add(adapter) [X](#), 668
 IlkRegistry.file(ilk, what, [data](#)) [X](#) [a](#), 669, 670
 IlkRegistry.put(_ilk, _join, _gem, _dec, _class, _pip, _xlip, _name, _symbol) [X](#) [a](#),
 673
 IlkRegistry.remove(ilk), 670
 IlkRegistry.update(ilk) [X](#), 673

definition

IlkRegistry.ilkData: [mapping](#)(bytes32 => IlkRegistry.Ilk), 667

read

IlkRegistry.add(adapter) [X](#), 668
 IlkRegistry.class(ilk), 671
 IlkRegistry.dec(ilk), 672
 IlkRegistry.file(ilk, what, [data](#)) [X](#) [a](#), 669, 670
 IlkRegistry.gem(ilk), 672
 IlkRegistry.info(ilk), 671
 IlkRegistry.join(ilk), 672
 IlkRegistry.name(ilk), 672
 IlkRegistry.pip(ilk), 672
 IlkRegistry.pos(ilk), 671
 IlkRegistry.put(_ilk, _join, _gem, _dec, _class, _pip, _xlip, _name, _symbol) [X](#) [a](#),
 673
 IlkRegistry.remove(ilk), 670
 IlkRegistry.remove(ilk) [X](#), 669
 IlkRegistry.symbol(ilk), 672
 IlkRegistry.update(ilk) [X](#), 673
 IlkRegistry.xlip(ilk), 672

IlkRegistry

definition

[contract](#) IlkRegistry, 667

ilks

call

Clipper.getFeedPrice(), 244
 ClipperMom.getPrices(clip), 258
 Clipper.upchost() [X](#), 248
 D3MHub.exec(ilk) [X](#), 324
 D3MInit._init(dss, d3m, cfg, gem), 725
 Dog.bark(ilk, urn, kpr) [X](#), 253
 DssAutoLine.exec(_ilk) [X](#), 707
 DssExecLib.decreaseIlkDebtCeiling(_ilk, _amount, _global) [X](#), 742
 DssExecLib.increaseIlkDebtCeiling(_ilk, _amount, _global) [X](#), 741
 DssExecLib.setIlkAutoLineDebtCeiling(_ilk, _amount) [X](#), 743
 DssExecLib.setIlkMinVaultAmount(_ilk, _amount) [X](#), 743
 DssProxyActions.draw(manager, jug, daiJoin, cdp, wad) [X](#), 536
 DssProxyActionsEnd.free(manager, end, cdp), 543
 DssProxyActions._getWipeAllWad(vat, usr, urn, ilk), 530
 DssProxyActions._getWipeDart(vat, dai, urn, ilk), 530
 DssProxyActions.lockETH(manager, ethJoin, cdp) [X](#), 534
 DssProxyActions.lockETHAndDraw(manager, jug, ethJoin, daiJoin, cdp, wadD) [X](#), 538

[DssProxyActions.lockGem\(manager, gemJoin, cdp, amt, transferFrom\)](#) [X](#), 535
[DssProxyActions.lockGemAndDraw\(manager, jug, gemJoin, daiJoin, cdp, amtC, wadD, transferFrom\)](#) [X](#), 539
[DssProxyActions.wipe\(manager, daiJoin, cdp, wad\)](#) [X](#), 537
[DssProxyActions.wipeAll\(manager, daiJoin, cdp\)](#) [X](#), 537
[DssProxyActions.wipeAllAndFreeETH\(manager, ethJoin, daiJoin, cdp, wadC\)](#) [X](#), 540
[DssProxyActions.wipeAllAndFreeGem\(manager, gemJoin, daiJoin, cdp, amtC\)](#) [X](#), 541
[DssProxyActions.wipeAndFreeETH\(manager, ethJoin, daiJoin, cdp, wadC, wadD\)](#) [X](#), 540
[DssProxyActions.wipeAndFreeGem\(manager, gemJoin, daiJoin, cdp, amtC, wadD\)](#) [X](#), 541
[End.cage\(ilk\)](#) [X](#), 299
[End.flow\(ilk\)](#) [X](#), 300
[End.skim\(ilk, urn\)](#) [X](#), 300
[End.skip\(ilk, id\)](#) [X](#), 299
[End.snip\(ilk, id\)](#) [X](#), 299
[IlkRegistry.add\(adapter\)](#) [X](#), 668
[IlkRegistry.update\(ilk\)](#) [X](#), 673
[Jug.drip\(ilk\)](#) [X](#), 275

write

[D3MHub.cage\(ilk\)](#) [X](#) [a](#), 325
[D3MHub.cull\(ilk\)](#) [X](#), 326
[D3MHub.file\(ilk, what, data\)](#) [X](#) [a](#), 321
[D3MHub.uncull\(ilk\)](#) [X](#), 326
[Dog.bark\(ilk, urn, kpr\)](#) [X](#), 253
[Dog.digs\(ilk, rad\)](#) [X](#) [a](#), 255
[Dog.file\(ilk, what, clip\)](#) [X](#) [a](#), 253
[Dog.file\(ilk, what, data\)](#) [X](#) [a](#), 253
[DssAutoLine.exec\(_ilk\)](#) [X](#), 707
[DssAutoLine.remIlk\(ilk\)](#) [X](#) [a](#), 706
[DssAutoLine.setIlk\(ilk, line, gap, ttl\)](#) [X](#) [a](#), 706
[DssCdpManager.open\(ilk, usr\)](#) [X](#), 522
[IlkRegistry.add\(adapter\)](#) [X](#), 668
[IlkRegistry.put\(_ilk, _join, _gem, _dec, _class, _pip, _xlip, _name, _symbol\)](#) [X](#) [a](#), 673
[IlkRegistry._remove\(ilk\)](#), 670
[Jug.drip\(ilk\)](#) [X](#), 275
[Jug.file\(ilk, what, data\)](#) [X](#) [a](#), 275
[Jug.init\(ilk\)](#) [X](#) [a](#), 274
[Spotter.file\(ilk, what, pip_\)](#) [X](#) [a](#), 228
[Spotter.file\(ilk, what, data\)](#) [X](#) [a](#), 228
[Vat.file\(ilk, what, data\)](#) [X](#) [a](#), 231
[Vat.fold\(i, u, rate\)](#) [X](#) [a](#), 234
[Vat.frob\(i, u, v, w, dink, dart\)](#) [X](#), 232
[Vat.grab\(i, u, v, w, dink, dart\)](#) [X](#) [a](#), 233
[Vat.init\(ilk\)](#) [X](#) [a](#), 231

definition

[D3MHub.ilks: mapping\(bytes32 => D3MHub.Ilk\)](#), 319
[Dog.ilks: mapping\(bytes32 => Dog.Ilk\)](#), 251
[DssAutoLine.ilks: mapping\(bytes32 => DssAutoLine.Ilk\)](#), 705
[DssCdpManager.ilks: mapping\(uint256 => bytes32\)](#), 521
[IlkRegistry.ilks: bytes32\[\], 667](#)
[Jug.ilks: mapping\(bytes32 => Jug.Ilk\)](#), 273
[Spotter.ilks: mapping\(bytes32 => Spotter.Ilk\)](#), 227
[Vat.ilks: mapping\(bytes32 => Vat.Ilk\)](#), 229

read

[D3MHub.cage\(ilk\)](#) [X](#) [a](#), 325
[D3MHub.cull\(ilk\)](#) [X](#), 326
[D3MHub.culled\(ilk\)](#), 327
[D3MHub.exec\(ilk\)](#) [X](#), 324
[D3MHub.exec\(ilk, _pool, Art, lineWad\)](#), 322
[D3MHub.exit\(ilk, usr, wad\)](#) [X](#), 325
[D3MHub.file\(ilk, what, data\)](#) [X](#) [a](#), 321
[D3MHub.plan\(ilk\)](#), 327
[D3MHub.pool\(ilk\)](#), 327

D3MHub.tau(ilk), 327
 D3MHub.tic(ilk), 327
 D3MHub.uncull(ilk) [X](#), 326
 Dog.bark(ilk, urn, kpr) [X](#), 253
 Dog.chop(ilk), 253
 Dog.digs(ilk, rad) [X](#) [a](#), 255
 Dog.file(ilk, what, clip) [X](#) [a](#), 253
 Dog.file(ilk, what, [data](#)) [X](#) [a](#), 253
 DssAutoLine.exec(_ilk) [X](#), 707
 DssAutoLine.remIlk(ilk) [X](#) [a](#), 706
 DssCdpManager.enter(src, cdp) [X](#), 525
 DssCdpManager.flux(cdp, dst, wad) [X](#), 524
 DssCdpManager.frob(cdp, dink, dart) [X](#), 524
 DssCdpManager.quit(cdp, dst) [X](#), 524
 DssCdpManager.shift(cdpSrc, cdpDst) [X](#), 525
 GetCdps.getCdpsAsc(manager, guy), 527
 GetCdps.getCdpsDesc(manager, guy), 527
 IlkRegistry.add(adapter) [X](#), 668
 IlkRegistry.count(), 670
 IlkRegistry.get(pos), 671
 IlkRegistry.list(), 670
 IlkRegistry.list(start, end), 671
 IlkRegistry.put(_ilk, _join, _gem, _dec, _class, _pip, _xlip, _name, _symbol) [X](#) [a](#),
 673
 IlkRegistry.remove(ilk), 670
 Jug.drip(ilk) [X](#), 275
 Jug.file(ilk, what, [data](#)) [X](#) [a](#), 275
 Jug.init(ilk) [X](#) [a](#), 274
 Spotter.file(ilk, what, pip_) [X](#) [a](#), 228
 Spotter.file(ilk, what, [data](#)) [X](#) [a](#), 228
 Spotter.poke(ilk) [X](#), 228
 Vat.file(ilk, what, [data](#)) [X](#) [a](#), 231
 Vat.fold(i, u, rate) [X](#) [a](#), 234
 Vat.fork(ilk, src, dst, dink, dart) [X](#), 233
 Vat.frob(i, u, v, w, dink, dart) [X](#), 232
 Vat.grab(i, u, v, w, dink, dart) [X](#) [a](#), 233
 Vat.init(ilk) [X](#) [a](#), 231
 ilkToChainlogFormat
 [call](#)
 D3MInit._init(dss, d3m, cfg, gem), 725
 [definition](#)
 ScriptTools.ilkToChainlogFormat(ilk), 716
 implementation
 [call](#)
 D3MCompoundPlan.active(), 317
 D3MCompoundPlan.[constructor](#)(cDai_) [X](#), 315
 D3MInit.initCompound(dss, d3m, cfg, compoundCfg), 727
 GemJoin6.[constructor](#)(vat_, ilk_, gem_) [X](#), 437
 GemJoin6.exit(usr, wad) [X](#), 438
 GemJoin6.join(usr, wad) [X](#), 438
 [write](#)
 TUSD.setImplementation(newImplementation) [X](#), 487
 [definition](#)
 TUSD.implementation: [address](#), 487
 implementations
 [write](#)
 AuthGemJoin8.setImplementation(implementation, permitted) [X](#) [a](#), 506
 GemJoin6.setImplementation(implementation, permitted) [X](#) [a](#), 438
 GemJoin7.setImplementation(implementation, permitted) [X](#) [a](#), 440
 GemJoin8.setImplementation(implementation, permitted) [X](#) [a](#), 442
 [definition](#)
 AuthGemJoin8.implementations: [mapping](#)([address](#) => [uint256](#)), 505
 GemJoin6.implementations: [mapping](#)([address](#) => [uint256](#)), 437

```

    GemJoin7.implementations: mapping(address => uint256), 439
    GemJoin8.implementations: mapping(address => uint256), 441
  read
    AuthGemJoin8.exit(usr, amt) X, 506
    AuthGemJoin8.join(urn, amt, msgSender) X a, 506
    GemJoin6.exit(usr, wad) X, 438
    GemJoin6.join(usr, wad) X, 438
    GemJoin7.exit(usr, amt) X, 440
    GemJoin7.join(usr, amt) X, 440
    GemJoin8.exit(usr, amt) X, 442
    GemJoin8.join(usr, amt) X, 442
inbox
  write
    L1DaiGateway.constructor(_inbox) [L1CrossDomainEnabled] X, 570
    L1GovernanceRelay.constructor(_inbox) [L1CrossDomainEnabled] X, 577
  definition
    L1DaiGateway.inbox [L1CrossDomainEnabled] : IInbox, 569
    L1GovernanceRelay.inbox [L1CrossDomainEnabled] : IInbox, 577
  read
    L1DaiGateway.modifier onlyL2Counterpart(l2Counterpart) [L1CrossDomainEnabled], 569
    L1DaiGateway.sendTxToL2(target, user, maxSubmissionCost, maxGas, gasPriceBid,
      data) [L1CrossDomainEnabled], 570
    L1DaiGateway.sendTxToL2NoAliasing(target, user, l1CallValue, maxSubmissionCost,
      maxGas, gasPriceBid, data) [L1CrossDomainEnabled], 570
    L1GovernanceRelay.modifier onlyL2Counterpart(l2Counterpart)
      [L1CrossDomainEnabled], 577
    L1GovernanceRelay.sendTxToL2(target, user, maxSubmissionCost, maxGas,
      gasPriceBid, data) [L1CrossDomainEnabled], 577
    L1GovernanceRelay.sendTxToL2NoAliasing(target, user, l1CallValue,
      maxSubmissionCost, maxGas, gasPriceBid, data) [L1CrossDomainEnabled], 578
increaseAllowance
  definition
    Dai.increaseAllowance(spender, addedValue) X, 589, 615
    DAIMock.increaseAllowance(spender, addedValue) [ERC20] X, 628
increaseGlobalDebtCeiling
  call
    DssExecLib.addNewCollateral(co) X, 750
    DssExecLib.increaseIlkDebtCeiling(_ilk, _amount, _global) X, 741
  definition
    DssExecLib.increaseGlobalDebtCeiling(_amount) X, 737
increaseIlkDebtCeiling
  definition
    DssExecLib.increaseIlkDebtCeiling(_ilk, _amount, _global) X, 741
info
  definition
    IlkRegistry.info(ilk), 671
init
  call
    D3MInit._init(dss, d3m, cfg, gem), 725
    D3MInit.initAave(dss, d3m, cfg, aaveCfg), 726
    D3MInit.initCompound(dss, d3m, cfg, compoundCfg), 727
    DssExecLib.addCollateralBase(_ilk, _gem, _join, _clip, _calc, _pip) X, 749
    MCD.initIlk(dss, ilk, join, pip), 720
  definition
    D3MInit._init(dss, d3m, cfg, gem), 725
    Jug.init(ilk) X a, 274
    Vat.init(ilk) X a, 231
initAave
  call
    D3MInitScript.run() X, 420
  definition
    D3MInit.initAave(dss, d3m, cfg, aaveCfg), 726
initCompound

```

call

D3MInitScript.run() **X**, 420

definition

D3MInit.initCompound(dss, d3m, cfg, compoundCfg), 727

initCore

call

D3MCoreInitScript.run() **X**, 374

definition

D3MInit.initCore(dss, d3mCore), 725

initialize

call

D3MCoreDeployScript.getChain(chainAlias) [StdChains], 343

D3MCoreDeployScript.getChain(chainId) [StdChains], 343

D3MCoreDeployScript.setChain(chainAlias, chain) [StdChains], 344

D3MCoreInitScript.getChain(chainAlias) [StdChains], 371

D3MCoreInitScript.getChain(chainId) [StdChains], 371

D3MCoreInitScript.setChain(chainAlias, chain) [StdChains], 372

D3MDeployScript.getChain(chainAlias) [StdChains], 397

D3MDeployScript.getChain(chainId) [StdChains], 397

D3MDeployScript.setChain(chainAlias, chain) [StdChains], 398

D3MInitScript.getChain(chainAlias) [StdChains], 417

D3MInitScript.getChain(chainId) [StdChains], 418

D3MInitScript.setChain(chainAlias, chain) [StdChains], 418

definition

D3MCoreDeployScript.initialize() [StdChains], 345

D3MCoreInitScript.initialize() [StdChains], 373

D3MDeployScript.initialize() [StdChains], 399

D3MInitScript.initialize() [StdChains], 419

initialized

write

D3MCoreDeployScript.initialize() [StdChains], 345

D3MCoreInitScript.initialize() [StdChains], 373

D3MDeployScript.initialize() [StdChains], 399

D3MInitScript.initialize() [StdChains], 419

definition

D3MCoreDeployScript.initialized [StdChains] : **bool**, 330

D3MCoreInitScript.initialized [StdChains] : **bool**, 358

D3MDeployScript.initialized [StdChains] : **bool**, 384

D3MInitScript.initialized [StdChains] : **bool**, 404

read

D3MCoreDeployScript.initialize() [StdChains], 345

D3MCoreInitScript.initialize() [StdChains], 373

D3MDeployScript.initialize() [StdChains], 399

D3MInitScript.initialize() [StdChains], 419

_initiateERC20Deposit

call

L1DAITokenBridge.depositERC20(_l1Token, _l2Token, _amount, _l2Gas, _data) **X**, 594

L1DAITokenBridge.depositERC20To(_l1Token, _l2Token, _to, _amount, _l2Gas, _data) **X**,
594

definition

L1DAITokenBridge._initiateERC20Deposit(_from, _to, _amount, _l2Gas, _data), 594

_initiateWithdrawal

call

L2DAITokenBridge.withdraw(_l2Token, _amount, _l1Gas, _data) **X**, 605

L2DAITokenBridge.withdrawTo(_l2Token, _to, _amount, _l1Gas, _data) **X**, 606

definition

L2DAITokenBridge._initiateWithdrawal(_from, _to, _amount, _l1Gas, _data), 606

initIlk

call

MCD.initIlk(dss, ilk), 720

MCD.initIlk(dss, ilk, join), 720

MCD.initIlk(dss, ilk, join, pip, clip, clipCalc), 720

definition

```

MCD.initIlk(dss, ilk), 720
MCD.initIlk(dss, ilk, join), 720
MCD.initIlk(dss, ilk, join, pip), 720
MCD.initIlk(dss, ilk, join, pip, clip, clipCalc), 720
INT256.MIN_ABS
  definition
    D3MCoreDeployScript.INT256.MIN_ABS [StdUtils] : uint256, 329
    D3MCoreInitScript.INT256.MIN_ABS [StdUtils] : uint256, 357
    D3MDeployScript.INT256.MIN_ABS [StdUtils] : uint256, 383
    D3MInitScript.INT256.MIN_ABS [StdUtils] : uint256, 403
  read
    D3MCoreDeployScript.bound(x, min, max) [StdUtils], 336
    D3MCoreInitScript.bound(x, min, max) [StdUtils], 364
    D3MDeployScript.bound(x, min, max) [StdUtils], 390
    D3MInitScript.bound(x, min, max) [StdUtils], 410
interestRateModel
  call
    D3MCompoundPlan.active(), 317
    D3MCompoundPlan.constructor(cDai_) X, 315
    D3MCompoundPlan.calculateTargetSupply(targetInterestRate, borrows), 316
    D3MInit.initCompound(dss, d3m, cfg, compoundCfg), 727
ipfs
  write
    ChainLog.setIPFS(_ipfs) X a, 710
  definition
    ChainLog.ipfs: string, 709
isAuthorized
  call
    ClipperMom.modifier auth(), 257
    D3MMom.modifier auth(), 355
    L1EscrowMom.modifier auth(), 637
  definition
    ClipperMom.isAuthorized(src, sig), 258
    D3MMom.isAuthorized(src, sig), 355
    L1EscrowMom.isAuthorized(src, sig), 637
isContract
  call
    Address.functionCallWithValue(target, data, value, errorMessage), 765
    Address.functionDelegateCall(target, data, errorMessage), 766
    Address.functionStaticCall(target, data, errorMessage), 765
    L1DAITokenBridge.depositERC20(_l1Token, _l2Token, _amount, _l2Gas, _data) X, 594
  definition
    Address.isContract(account), 763
isFork
  call
    D3MCoreDeployScript.modifier skipWhenForking() [StdCheatsSafe], 334
    D3MCoreDeployScript.modifier skipWhenNotForking() [StdCheatsSafe], 335
    D3MCoreInitScript.modifier skipWhenForking() [StdCheatsSafe], 362
    D3MCoreInitScript.modifier skipWhenNotForking() [StdCheatsSafe], 363
    D3MDeployScript.modifier skipWhenForking() [StdCheatsSafe], 388
    D3MDeployScript.modifier skipWhenNotForking() [StdCheatsSafe], 389
    D3MInitScript.modifier skipWhenForking() [StdCheatsSafe], 409
    D3MInitScript.modifier skipWhenNotForking() [StdCheatsSafe], 409
  definition
    D3MCoreDeployScript.isFork() [StdCheatsSafe], 343
    D3MCoreInitScript.isFork() [StdCheatsSafe], 371
    D3MDeployScript.isFork() [StdCheatsSafe], 397
    D3MInitScript.isFork() [StdCheatsSafe], 417
isOpen
  write
    L1DAIBridge.close() X a, 618
    L1DaiGateway.close() X a, 571
    L1DAITokenBridge.close() X a, 594

```


L2DaiGateway.close() **X** **a**, 582

L2DAITokenBridge.close() **X** **a**, 605

definition

L1DAIBridge.isOpen: **uint256**, 617

L1DaiGateway.isOpen: **uint256**, 569

L1DAITokenBridge.isOpen: **uint256**, 591

L2DaiGateway.isOpen: **uint256**, 581

L2DAITokenBridge.isOpen: **uint256**, 603

read

L1DAIBridge.modifier whenOpen(), 618

L1DaiGateway.outboundTransfer(l1Token, to, amount, maxGas, gasPriceBid, **data**) **X**, 571

L1DAITokenBridge.initiateERC20Deposit(_from, _to, _amount, _l2Gas, _data), 594

L2DaiGateway.outboundTransfer(l1Token, to, amount, , , **data**) **X**, 583

L2DAITokenBridge.initiateWithdrawal(_from, _to, _amount, _l1Gas, _data), 606

IS_SCRIPT

definition

D3MCoreDeployScript.IS_SCRIPT [Script] : **bool**, 329

D3MCoreInitScript.IS_SCRIPT [Script] : **bool**, 357

D3MDeployScript.IS_SCRIPT [Script] : **bool**, 383

D3MInitScript.IS_SCRIPT [Script] : **bool**, 403

isStopped

call

Clipper.kick(tab, lot, usr, kpr) **X** **a**, 244

Clipper.redo(id, kpr) **X**, 245

Clipper.take(id, amt, max, who, **data**) **X**, 246

definition

Clipper.modifier isStopped(level), 242

isValid

call

TeleportOracleAuth.requestMint(teleportGUID, signatures, maxFeePercentage, operatorFee) **X**, 650

definition

TeleportOracleAuth.isValid(signHash, signatures, threshold_), 650

J

join

call

CalleeMakerOtcDai.clipperCall([sender](#), daiAmt, gemAmt, [data](#)) [X](#), 550
 CurveLpTokenUniv3Callee.clipperCall([sender](#), owe, slice, [data](#)) [X](#), 552
 D3MHub._exec(ilk, _pool, Art, lineWad), 322
 D3MHub.wipe(ilk, _pool, urn), 322
 DssProxyActions.daiJoin_join(apt, urn, wad) [Common] [X](#), 529
 DssProxyActionsDsr.daiJoin_join(apt, urn, wad) [Common] [X](#), 547
 DssProxyActionsDsr.join(daiJoin, pot, wad) [X](#), 547
 DssProxyActionsEnd.daiJoin_join(apt, urn, wad) [Common] [X](#), 543
 DssProxyActions.ethJoin_join(apt, urn) [X](#), 531
 DssProxyActions.gemJoin_join(apt, urn, amt, transferFrom) [X](#), 531
 DssPsm.buyGem(usr, gemAmt) [X](#), 511
 DssPsm.sellGem(usr, gemAmt) [X](#), 510
 GemJoin9.join(usr) [X](#), 444
 GemJoin9.join(usr, wad) [X](#), 444
 PSMCallee.clipperCall([sender](#), owe, slice, [data](#)) [X](#), 555
 TeleportJoin.settle(sourceDomain, batchedDaiToFlush) [X](#), 645
 UniswapV2CalleeDai.clipperCall([sender](#), daiAmt, gemAmt, [data](#)) [X](#), 558
 UniswapV2LpTokenCalleeDai.clipperCall([sender](#), daiAmt, gemAmt, [data](#)) [X](#), 560
 UniswapV3Callee.clipperCall([sender](#), owe, slice, [data](#)) [X](#), 567
 WstETHCurveUniv3Callee.clipperCall([sender](#), owe, slice, [data](#)) [X](#), 564

write

IlkRegistry.file(ilk, what, [data](#)) [X](#) [a](#), 669

definition

AuthGemJoin5.join(urn, amt, msgSender) [X](#) [a](#), 504
 AuthGemJoin8.join(urn, amt, msgSender) [X](#) [a](#), 506
 AuthGemJoin.join(urn, wad, msgSender) [X](#) [a](#), 508
 AuthGemJoin.join(usr, wad) [X](#) [a](#), 448
 DaiJoin.join(usr, wad) [X](#), 426
 DssProxyActionsDsr.join(daiJoin, pot, wad) [X](#), 547
 ESM.join(wad) [X](#), 305
 GemJoin2.join(usr, wad) [X](#), 428
 GemJoin3.join(usr, amt) [X](#), 430
 GemJoin4.join(usr, wad) [X](#), 434
 GemJoin5.join(usr, amt) [X](#), 436
 GemJoin6.join(usr, wad) [X](#), 438
 GemJoin7.join(usr, amt) [X](#), 440
 GemJoin8.join(usr, amt) [X](#), 442
 GemJoin9._join(usr), 444
 GemJoin9.join(usr) [X](#), 444
 GemJoin9.join(usr, wad) [X](#), 444
 GemJoin.join(usr, wad) [X](#), 424
 IlkRegistry.join(ilk), 672
 ManagedGemJoin.join(usr, amt) [X](#) [a](#), 450
 Pot.join(wad) [X](#), 279

read

IlkRegistry.add(adapter) [X](#), 668
 IlkRegistry.join(ilk), 672
 IlkRegistry.remove(ilk) [X](#), 669
 IlkRegistry.update(ilk) [X](#), 673

Jug

definition

[contract](#) Jug, 273

jug

call

DssExecLib.accumulateCollateralStabilityFees(_ilk) [X](#), 735
 DssExecLib.addCollateralBase(_ilk, _gem, _join, _clip, _calc, _pip) [X](#), 749
 DssExecLib.setGlobalStabilityFee(_rate) [X](#), 741
 DssExecLib.setIlkStabilityFee(_ilk, _rate, _doDrip) [X](#), 746

definition

DssExecLib.jug(), 731

jumpMultiplierPerBlock

[call](#)

D3MCompoundPlan._calculateTargetSupply(targetInterestRate, borrows), 316

K

keys

write

ChainLog.addAddress(_key, _addr), 711
 ChainLog.removeAddress(_key), 711

definition

ChainLog.keys: `bytes32[]`, 709

read

ChainLog.addAddress(_key, _addr), 711
 ChainLog.count(), 711
 ChainLog.get(_index), 711
 ChainLog.list(), 711
 ChainLog.removeAddress(_key), 711
 ChainLog.setAddress(_key, _addr) **X** **a**, 710

kick

call

Dog.bark(ilk, urn, kpr) **X**, 253
 Vow.flap() **X**, 271
 Vow.flop() **X**, 271

definition

Clipper.kick(tab, lot, usr, kpr) **X** **a**, 244
 Flapper.kick(lot, bid) **X** **a**, 263
 Flopper.kick(gal, lot, bid) **X** **a**, 266

kicks

write

Clipper.kick(tab, lot, usr, kpr) **X** **a**, 244
 Flapper.kick(lot, bid) **X** **a**, 263
 Flopper.kick(gal, lot, bid) **X** **a**, 266

definition

Clipper.kicks: `uint256`, 241
 Flapper.kicks: `uint256`, 261
 Flopper.kicks: `uint256`, 265

read

Clipper.kick(tab, lot, usr, kpr) **X** **a**, 244
 Flapper.kick(lot, bid) **X** **a**, 263
 Flopper.kick(gal, lot, bid) **X** **a**, 266

king

write

D3MAavePool.file(what, **data**) **X** **a**, 350
 D3MCompoundPool.file(what, **data**) **X** **a**, 378

definition

D3MAavePool.king: `address`, 349
 D3MCompoundPool.king: `address`, 377

read

D3MAavePool.collect() **X**, 352
 D3MCompoundPool.collect(claim) **X**, 380

kink

call

D3MCompoundPlan.calculateTargetSupply(targetInterestRate, borrows), 316

kiss

call

DssExecLib.addReaderToWhitelist(_oracle, _reader) **X**, 747
 Flopper.dent(id, lot, bid) **X**, 267

definition

Median.kiss(a) **X** **a**, 284
 OSM.kiss(a) **X** **a**, 288
 TrustedRelay.kiss(usr) **X** **a**, 662
 Vow.kiss(rad) **X**, 271

KNC

definition

contract KNC, 465

L**L1Counterpart****write**

L2DaiGateway.**constructor**(_l1Counterpart, _l2Router, _l1Dai, _l2Dai) **X**, 582

definition

L2DaiGateway.l1Counterpart: **address**, 581

read

L2DaiGateway.counterpartGateway(), 584

L2DaiGateway.finalizeInboundTransfer(l1Token, from, to, amount,) **X**, 583

L2DaiGateway.outboundTransfer(l1Token, to, amount, , , **data**) **X**, 583

L1Dai**write**

L1DaiGateway.**constructor**(_l2Counterpart, _l1Router, _inbox, _l1Dai, _l2Dai, _l1Escrow) **X**, 571

L2DaiGateway.**constructor**(_l1Counterpart, _l2Router, _l1Dai, _l2Dai) **X**, 582

definition

L1DaiGateway.l1Dai: **address**, 569

L2DaiGateway.l1Dai: **address**, 581

read

L1DaiGateway.calculateL2TokenAddress(l1Token), 573

L1DaiGateway.finalizeInboundTransfer(l1Token, from, to, amount, **data**) **X**, 572

L1DaiGateway.outboundTransfer(l1Token, to, amount, maxGas, gasPriceBid, **data**) **X**, 571

L2DaiGateway.calculateL2TokenAddress(l1Token), 584

L2DaiGateway.finalizeInboundTransfer(l1Token, from, to, amount,) **X**, 583

L2DaiGateway.outboundTransfer(l1Token, to, amount, , , **data**) **X**, 583

L1DAIBridge**definition**

contract L1DAIBridge, 617

L1DaiGateway**definition**

contract L1DaiGateway, 569

L1DAITeleportGateway**definition**

contract L1DAITeleportGateway, 623

L1DAITokenBridge**definition**

contract L1DAITokenBridge, 591

L1DAITokenBridge**write**

L2DAITokenBridge.**constructor**(_l2CrossDomainMessenger, _l2Token, _l1Token, _l1DAITokenBridge) **X**, 605

definition

L2DAITokenBridge.l1DAITokenBridge: **address**, 603

read

L2DAITokenBridge.finalizeDeposit(_l1Token, _l2Token, _from, _to, _amount, _data) **X**, 606

L2DAITokenBridge._initiateWithdrawal(_from, _to, _amount, _l1Gas, _data), 606

L1Escrow**definition**

contract L1Escrow, 575

contract L1Escrow(2), 597

contract L1Escrow(3), 635

L1Escrow**write**

L1DaiGateway.**constructor**(_l2Counterpart, _l1Router, _inbox, _l1Dai, _l2Dai, _l1Escrow) **X**, 571

L1DAITeleportGateway.**constructor**(_starkNet, _l1Token, _l2TeleportGateway, _l1Escrow, _l1TeleportRouter) **X**, 623

definition

L1DaiGateway.l1Escrow: **address**, 569

L1DAITeleportGateway.l1Escrow: **address**, 623

read

L1DaiGateway.finalizeInboundTransfer(l1Token, from, to, amount, [data](#)) [X](#), 572
L1DaiGateway.outboundTransfer(l1Token, to, amount, maxGas, gasPriceBid, [data](#)) [X](#), 571
L1DAITeleportGateway.finalizeFlush(targetDomain, daiToFlush) [X](#), 623

L1EscrowMom
[definition](#)
[contract](#) L1EscrowMom, 637

L1GovernanceRelay
[definition](#)
[contract](#) L1GovernanceRelay, 577
[contract](#) L1GovernanceRelay(2), 599
[contract](#) L1GovernanceRelay(3), 621

l1GovernanceRelay
[write](#)
L2GovernanceRelay.[constructor](#)(_l1GovernanceRelay) [X](#), 585
L2GovernanceRelay.[constructor](#)(_l2CrossDomainMessenger, _l1GovernanceRelay) [X](#), 610
[definition](#)
L2GovernanceRelay.l1GovernanceRelay: [address](#), 585, 609
[read](#)
L2GovernanceRelay.relay(target, targetData) [X](#), 586, 611

l1Router
[write](#)
L1DaiGateway.[constructor](#)(_l2Counterpart, _l1Router, _inbox, _l1Dai, _l2Dai, _l1Escrow) [X](#), 571
[definition](#)
L1DaiGateway.l1Router: [address](#), 569
[read](#)
L1DaiGateway.parseOutboundData([data](#)), 572

l1TeleportRouter
[write](#)
L1DAITeleportGateway.[constructor](#)(_starkNet, _l1Token, _l2TeleportGateway, _l1Escrow, _l1TeleportRouter) [X](#), 623
[definition](#)
L1DAITeleportGateway.l1TeleportRouter: [TeleportRouter](#), 623
[read](#)
L1DAITeleportGateway.finalizeFlush(targetDomain, daiToFlush) [X](#), 623
L1DAITeleportGateway.finalizeRegisterTeleport(teleport) [X](#), 623

l1Token
[write](#)
L1DAITeleportGateway.[constructor](#)(_starkNet, _l1Token, _l2TeleportGateway, _l1Escrow, _l1TeleportRouter) [X](#), 623
L1DAITokenBridge.[constructor](#)(_l1Token, _l2DAITokenBridge, _l2Token, _l1messenger, _escrow) [X](#), 593
L2DAITokenBridge.[constructor](#)(_l2CrossDomainMessenger, _l2Token, _l1Token, _l1DAITokenBridge) [X](#), 605
[definition](#)
L1DAITeleportGateway.l1Token: [address](#), 623
L1DAITokenBridge.l1Token: [address](#), 591
L2DAITokenBridge.l1Token: [address](#), 603
[read](#)
L1DAITeleportGateway.finalizeFlush(targetDomain, daiToFlush) [X](#), 623
L1DAITokenBridge.depositERC20(_l1Token, _l2Token, _amount, _l2Gas, _data) [X](#), 594
L1DAITokenBridge.depositERC20To(_l1Token, _l2Token, _to, _amount, _l2Gas, _data) [X](#), 594
L1DAITokenBridge.finalizeERC20Withdrawal(_l1Token, _l2Token, _from, _to, _amount, _data) [X](#), 595
L1DAITokenBridge._initiateERC20Deposit(_from, _to, _amount, _l2Gas, _data), 594
L2DAITokenBridge.finalizeDeposit(_l1Token, _l2Token, _from, _to, _amount, _data) [X](#), 606
L2DAITokenBridge._initiateWithdrawal(_from, _to, _amount, _l1Gas, _data), 606

l2Counterpart
[write](#)

```

L1DaiGateway.constructor(_l2Counterpart, _l1Router, _inbox, _l1Dai, _l2Dai,
    _l1Escrow) X, 571
definition
L1DaiGateway.l2Counterpart: address, 569
read
L1DaiGateway.counterpartGateway(), 573
L1DaiGateway.finalizeInboundTransfer(l1Token, from, to, amount, data) X, 572
L1DaiGateway.outboundTransfer(l1Token, to, amount, maxGas, gasPriceBid, data) X,
    571
l2Dai
write
L1DAIBridge.constructor(_starkNet, _dai, _l2Dai, _escrow, _l2DaiBridge) X, 618
L1DaiGateway.constructor(_l2Counterpart, _l1Router, _inbox, _l1Dai, _l2Dai,
    _l1Escrow) X, 571
L2DaiGateway.constructor(_l1Counterpart, _l2Router, _l1Dai, _l2Dai) X, 582
definition
L1DAIBridge.l2Dai: uint256, 617
L1DaiGateway.l2Dai: address, 569
L2DaiGateway.l2Dai: address, 581
read
L1DAIBridge.deposit(amount, l2Recipient) X, 619
L1DaiGateway.calculateL2TokenAddress(l1Token), 573
L2DaiGateway.calculateL2TokenAddress(l1Token), 584
L2DaiGateway.finalizeInboundTransfer(l1Token, from, to, amount, ) X, 583
L2DaiGateway.outboundTransfer(l1Token, to, amount, , , data) X, 583
l2DaiBridge
write
L1DAIBridge.constructor(_starkNet, _dai, _l2Dai, _escrow, _l2DaiBridge) X, 618
definition
L1DAIBridge.l2DaiBridge: uint256, 617
read
L1DAIBridge.cancelDeposit(amount, l2Recipient, l1Recipient, nonce) X, 620
L1DAIBridge.deposit(amount, l2Recipient) X, 619
L1DAIBridge.forceWithdrawal(amount, l2Sender) X, 620
L1DAIBridge.startDepositCancellation(amount, l2Recipient, nonce) X, 620
L1DAIBridge.withdraw(amount, l1Recipient) X, 619
L2DaiGateway
definition
contract L2DaiGateway, 581
L2DAITokenBridge
definition
contract L2DAITokenBridge, 603
l2DAITokenBridge
write
L1DAITokenBridge.constructor(_l1Token, _l2DAITokenBridge, _l2Token, _l1messenger,
    _escrow) X, 593
definition
L1DAITokenBridge.l2DAITokenBridge: address, 591
read
L1DAITokenBridge.finalizeERC20Withdrawal(_l1Token, _l2Token, _from, _to, _amount,
    _data) X, 595
L1DAITokenBridge._initiateERC20Deposit(_from, _to, _amount, _l2Gas, _data), 594
L2GovernanceRelay
definition
contract L2GovernanceRelay, 585
contract L2GovernanceRelay(2), 609
l2GovernanceRelay
write
L1GovernanceRelay.constructor(_inbox, _l2GovernanceRelay) X, 579
L1GovernanceRelay.constructor(_l2GovernanceRelay, _l1messenger) X, 601
L1GovernanceRelay.constructor(_starkNet, _l2GovernanceRelay) X, 621
definition
L1GovernanceRelay.l2GovernanceRelay: address, 577, 599

```



```

    L1GovernanceRelay.l2GovernanceRelay:  uint256, 621
read
    L1GovernanceRelay.relay(spell) X a, 621
    L1GovernanceRelay.relay(target, targetData, l1CallValue, maxGas, gasPriceBid,
        maxSubmissionCost) X a, 579
    L1GovernanceRelay.relay(target, targetData, l2gas) X a, 601
l2Router
    write
        L2DaiGateway.constructor(_l1Counterpart, _l2Router, _l1Dai, _l2Dai) X, 582
    definition
        L2DaiGateway.l2Router:  address, 581
    read
        L2DaiGateway.parseOutboundData(data), 584
l2TeleportGateway
    write
        L1DAITeleportGateway.constructor(_starkNet, _l1Token, _l2TeleportGateway,
            _l1Escrow, _l1TeleportRouter) X, 623
    definition
        L1DAITeleportGateway.l2TeleportGateway:  uint256, 623
    read
        L1DAITeleportGateway.finalizeFlush(targetDomain, daiToFlush) X, 623
        L1DAITeleportGateway.finalizeRegisterTeleport(teleport) X, 623
l2Token
    write
        L1DAITokenBridge.constructor(_l1Token, _l2DAITokenBridge, _l2Token, _l1messenger,
            _escrow) X, 593
        L2DAITokenBridge.constructor(_l2CrossDomainMessenger, _l2Token, _l1Token,
            _l1DAITokenBridge) X, 605
    definition
        L1DAITokenBridge.l2Token:  address, 591
        L2DAITokenBridge.l2Token:  address, 603
    read
        L1DAITokenBridge.depositERC20(_l1Token, _l2Token, _amount, _l2Gas, _data) X, 594
        L1DAITokenBridge.depositERC20To(_l1Token, _l2Token, _to, _amount, _l2Gas, _data) X,
            594
        L1DAITokenBridge.finalizeERC20Withdrawal(_l1Token, _l2Token, _from, _to, _amount,
            _data) X, 595
        L1DAITokenBridge._initiateERC20Deposit(_from, _to, _amount, _l2Gas, _data), 594
        L2DAITokenBridge.finalizeDeposit(_l1Token, _l2Token, _from, _to, _amount, _data) X,
            606
        L2DAITokenBridge._initiateWithdrawal(_from, _to, _amount, _l1Gas, _data), 606
        L2DAITokenBridge.withdraw(_l2Token, _amount, _l1Gas, _data) X, 605
        L2DAITokenBridge.withdrawTo(_l2Token, _to, _amount, _l1Gas, _data) X, 606
l2ToL1Sender
    call
        L1DaiGateway.modifier onlyL2Counterpart(l2Counterpart) [L1CrossDomainEnabled], 569
        L1GovernanceRelay.modifier onlyL2Counterpart(l2Counterpart)
            [L1CrossDomainEnabled], 577
label
    call
        D3MCoreDeployScript.makeAddrAndKey(name) [StdCheatsSafe], 342
        D3MCoreInitScript.makeAddrAndKey(name) [StdCheatsSafe], 370
        D3MDeployScript.makeAddrAndKey(name) [StdCheatsSafe], 396
        D3MInitScript.makeAddrAndKey(name) [StdCheatsSafe], 417
lad
    write
        GemBag.constructor(lad_, gem_) X, 431
    definition
        GemBag.lad:  address, 431
    read
        GemBag.exit(usr, wad) X, 431
last
    write

```

```

DssAutoLine.exec(_ilk) X, 707
DssCdpManager.give(cdp, dst) X, 523
DssCdpManager.open(ilk, usr) X, 522
definition
DssCdpManager.last: mapping(address => uint256), 521
read
DssAutoLine.exec(_ilk) X, 707
DssCdpManager.give(cdp, dst) X, 523
DssCdpManager.open(ilk, usr) X, 522
GetCdps.getCdpsDesc(manager, guy), 527
lastInc
write
DssAutoLine.exec(_ilk) X, 707
read
DssAutoLine.exec(_ilk) X, 707
lCount
write
Cure.load(src) X, 293
definition
Cure.lCount: uint256, 291
read
Cure.load(src) X, 293
Cure.tell(), 292
length
call
EnumerableSet.length(set), 758, 759, 761
TeleportRouter.numDomains(), 654
definition
EnumerableSet.length(set), 758, 759, 761
EnumerableSet._length(set), 757
lerpFab
call
DssExecLib.linearInterpolation(_name, _target, _ilk, _what, _startTime, _start,
    _end, _duration) X, 752
DssExecLib.linearInterpolation(_name, _target, _what, _startTime, _start, _end,
    _duration) X, 751
definition
DssExecLib.lerpFab(), 732
lid
write
Flapper.file(what, data) X a, 263
definition
Flapper.lid: uint256, 261
read
Flapper.kick(lot, bid) X a, 263
lift
definition
Cure.lift(src) X a, 293
Median.lift(a) X a, 283
limited
call
DssAction.execute() X, 307
definition
DssAction.modifier limited(), 307
Line
call
D3MHub._exec(ilk, _pool, Art, lineWad), 322
D3MInit._init(dss, d3m, cfg, gem), 725
DssAutoLine.exec(_ilk) X, 707
DssExecLib.decreaseGlobalDebtCeiling(_amount) X, 737
DssExecLib.increaseGlobalDebtCeiling(_amount) X, 737
write
Vat.file(what, data) X a, 231

```

```

definition
    Vat.Line: uint256, 229
read
    Vat.frob(i, u, v, w, dink, dart) X, 232
line
write
    TeleportJoin.file(what, domain_, data) X a, 642
    Vat.file(ilk, what, data) X a, 231
definition
    TeleportJoin.line: mapping(bytes32 => uint256), 641
read
    DssAutoLine.exec(_ilk) X, 707
    TeleportJoin._mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643
LinearDecrease
definition
    contract LinearDecrease, 235
linearInterpolation
definition
    DssExecLib.linearInterpolation(name, _target, _ilk, _what, _startTime, _start,
        _end, _duration) X, 752
    DssExecLib.linearInterpolation(name, _target, _what, _startTime, _start, _end,
        _duration) X, 751
LINK
definition
    contract LINK, 467
List
definition
    struct DssCdpManager.List, 521
list
write
    DssCdpManager.give(cdp, dst) X, 523
    DssCdpManager.open(ilk, usr) X, 522
definition
    ChainLog.list(), 711
    Clipper.list(), 248
    Cure.list(), 292
    DssCdpManager.list: mapping(uint256 => DssCdpManager.List), 521
    IlkRegistry.list(), 670
    IlkRegistry.list(start, end), 671
read
    DssCdpManager.give(cdp, dst) X, 523
    DssCdpManager.open(ilk, usr) X, 522
    GetCdps.getCdpsAsc(manager, guy), 527
    GetCdps.getCdpsDesc(manager, guy), 527
live
call
    D3MAavePool.file(what, data) X a, 350
    D3MAavePool.quit(dst) X a, 351
    D3MCompoundPool.file(what, data) X a, 378
    D3MCompoundPool.quit(dst) X a, 379
    D3MHub.cage(ilk) X a, 325
    D3MHub.cull(ilk) X, 326
    D3MHub.exec(ilk) X, 324
    D3MHub.file(ilk, what, data) X a, 321
    D3MHub.file(what, data) X a, 321
    D3MHub.uncull(ilk) X, 326
    D3MOracle.file(what, data) X a, 354
    D3MOracle.peek(), 354
    DssVestSuckable.pay(_guy, _amt), 693
    ESM.join(wad) X, 305
    IlkRegistry.constructor(vat_, dog_, cat_, spot_) X, 668
    IlkRegistry.remove(ilk) X, 669
    IlkRegistry.update(ilk) X, 673

```

TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643
 TeleportJoin.settle(sourceDomain, batchedDaiToFlush) **X**, 645

write

AuthGemJoin5.constructor(vat_, ilk_, gem_) **X**, 503
 AuthGemJoin5.cage() **X** **a**, 504
 AuthGemJoin8.constructor(vat_, ilk_, gem_) **X**, 505
 AuthGemJoin8.cage() **X** **a**, 506
 AuthGemJoin.constructor(vat_, ilk_, gem_) **X**, 447, 507
 AuthGemJoin.cage() **X** **a**, 447, 507
 Cure.constructor() **X**, 292
 Cure.cage() **X** **a**, 293
 DaiJoin.constructor(vat_, dai_) **X**, 425
 DaiJoin.cage() **X** **a**, 425
 Dog.constructor(vat_) **X**, 252
 Dog.cage() **X** **a**, 255
 End.constructor() **X**, 297
 End.cage() **X** **a**, 299
 ESM.constructor(gem_, end_, proxy_, min_) **X**, 303
 ESM.cage() **X** **a**, 304
 Flapper.constructor(vat_, gem_) **X**, 262
 Flapper.cage(rad) **X** **a**, 264
 Flopper.constructor(vat_, gem_) **X**, 266
 Flopper.cage() **X** **a**, 267
 GemJoin2.constructor(vat_, ilk_, gem_) **X**, 427
 GemJoin2.cage() **X** **a**, 427
 GemJoin3.constructor(vat_, ilk_, gem_, decimals) **X**, 429
 GemJoin3.cage() **X** **a**, 429
 GemJoin4.constructor(vat_, ilk_, gem_) **X**, 433
 GemJoin4.cage() **X** **a**, 433
 GemJoin5.constructor(vat_, ilk_, gem_) **X**, 435
 GemJoin5.cage() **X** **a**, 435
 GemJoin6.constructor(vat_, ilk_, gem_) **X**, 437
 GemJoin6.cage() **X** **a**, 438
 GemJoin7.constructor(vat_, ilk_, gem_) **X**, 439
 GemJoin7.cage() **X** **a**, 440
 GemJoin8.constructor(vat_, ilk_, gem_) **X**, 441
 GemJoin8.cage() **X** **a**, 442
 GemJoin9.constructor(vat_, ilk_, gem_) **X**, 443
 GemJoin9.cage() **X** **a**, 444
 GemJoin.constructor(vat_, ilk_, gem_) **X**, 424
 GemJoin.cage() **X** **a**, 424
 ManagedGemJoin.constructor(vat_, ilk_, gem_) **X**, 449
 ManagedGemJoin.cage() **X** **a**, 450
 Pot.constructor(vat_) **X**, 277
 Pot.cage() **X** **a**, 279
 Spotter.constructor(vat_) **X**, 227
 Spotter.cage() **X** **a**, 228
 Vat.constructor() **X**, 230
 Vat.cage() **X** **a**, 231
 Vow.constructor(vat_, flapper_, flopper_) **X**, 269
 Vow.cage() **X** **a**, 271

definition

AuthGemJoin5.live: **uint256**, 503
 AuthGemJoin8.live: **uint256**, 505
 AuthGemJoin.live: **uint256**, 447, 507
 Cure.live: **uint256**, 291
 DaiJoin.live: **uint256**, 425
 Dog.live: **uint256**, 251
 End.live: **uint256**, 295
 ESM.live: **uint256**, 303
 Flapper.live: **uint256**, 261
 Flopper.live: **uint256**, 265
 GemJoin2.live: **uint256**, 427

GemJoin3.live: [uint256](#), 429
GemJoin4.live: [uint256](#), 433
GemJoin5.live: [uint256](#), 435
GemJoin6.live: [uint256](#), 437
GemJoin7.live: [uint256](#), 439
GemJoin8.live: [uint256](#), 441
GemJoin9.live: [uint256](#), 443
GemJoin.live: [uint256](#), 423
ManagedGemJoin.live: [uint256](#), 449
Pot.live: [uint256](#), 277
Spotter.live: [uint256](#), 227
Vat.live: [uint256](#), 229
Vow.live: [uint256](#), 269

read

AuthGemJoin5.join(urn, amt, msgSender) [X](#) [a](#), 504
AuthGemJoin8.join(urn, amt, msgSender) [X](#) [a](#), 506
AuthGemJoin.join(urn, wad, msgSender) [X](#) [a](#), 508
AuthGemJoin.join(usr, wad) [X](#) [a](#), 448
Cure.cage() [X](#) [a](#), 293
Cure.deny(usr) [X](#) [a](#), 292
Cure.drop(src) [X](#) [a](#), 293
Cure.file(what, [data](#)) [X](#) [a](#), 292
Cure.lift(src) [X](#) [a](#), 293
Cure.load(src) [X](#), 293
Cure.rely(usr) [X](#) [a](#), 292
Cure.tell(), 292
DaiJoin.exit(usr, wad) [X](#), 426
Dog.bark(ilk, urn, kpr) [X](#), 253
End.cage() [X](#) [a](#), 299
End.cage(ilk) [X](#), 299
End.file(what, [data](#)) [X](#) [a](#), 298
End.free(ilk) [X](#), 300
End.thaw() [X](#), 300
ESM.denyProxy(target) [X](#), 305
ESM.fire() [X](#), 304
ESM.join(wad) [X](#), 305
Flapper.deal(id) [X](#), 264
Flapper.kick(lot, bid) [X](#) [a](#), 263
Flapper.tend(id, lot, bid) [X](#), 263
Flapper.yank(id) [X](#), 264
Flopper.deal(id) [X](#), 267
Flopper.dent(id, lot, bid) [X](#), 267
Flopper.kick(gal, lot, bid) [X](#) [a](#), 266
Flopper.yank(id) [X](#), 267
GemJoin2.join(usr, wad) [X](#), 428
GemJoin3.join(usr, amt) [X](#), 430
GemJoin4.join(usr, wad) [X](#), 434
GemJoin5.join(usr, amt) [X](#), 436
GemJoin6.join(usr, wad) [X](#), 438
GemJoin7.join(usr, amt) [X](#), 440
GemJoin8.join(usr, amt) [X](#), 442
GemJoin9._join(usr), 444
GemJoin.join(usr, wad) [X](#), 424
ManagedGemJoin.join(usr, amt) [X](#) [a](#), 450
Pot.file(what, [data](#)) [X](#) [a](#), 278
Spotter.file(ilk, what, pip-) [X](#) [a](#), 228
Spotter.file(ilk, what, [data](#)) [X](#) [a](#), 228
Spotter.file(what, [data](#)) [X](#) [a](#), 228
Vat.deny(usr) [X](#) [a](#), 229
Vat.file(ilk, what, [data](#)) [X](#) [a](#), 231
Vat.file(what, [data](#)) [X](#) [a](#), 231
Vat.fold(i, u, rate) [X](#) [a](#), 234
Vat.frob(i, u, v, w, dink, dart) [X](#), 232

Vat.rely(usr) [X](#) [a](#), 229

Vow.cage() [X](#) [a](#), 271

Vow.rely(usr) [X](#) [a](#), 269

load

call

GodMode.setBalance(token, who, amount), 768

GodMode.setWard(base, target, val), 767

definition

Cure.load(src) [X](#), 293

loadConfig

call

D3MCoreDeployScript.run() [X](#), 346

D3MCoreInitScript.run() [X](#), 374

D3MDeployScript.run() [X](#), 400

D3MInitScript.run() [X](#), 420

definition

ScriptTools.loadConfig(), 714

ScriptTools.loadConfig(name), 714

loadDependencies

call

D3MCoreInitScript.run() [X](#), 374

D3MDeployScript.run() [X](#), 400

D3MInitScript.run() [X](#), 420

OptionalLoadDependencies.loadDependencies(name) [X](#), 381

definition

OptionalLoadDependencies.loadDependencies(name) [X](#), 381

ScriptTools.loadDependencies(), 715

ScriptTools.loadDependencies(name), 714

loaded

write

Cure.load(src) [X](#), 293

definition

Cure.loaded: [mapping](#)(address => uint256), 291

read

Cure.load(src) [X](#), 293

loadFromChainlog

call

D3MCoreDeployScript.run() [X](#), 346

D3MCoreInitScript.run() [X](#), 374

D3MDeployScript.run() [X](#), 400

D3MInitScript.run() [X](#), 420

MCD.loadFromChainlog(chainlog), 719

definition

MCD.loadFromChainlog(chainlog), 719

Location

definition

[struct](#) ChainLog.Location, 709

location

write

ChainLog.addAddress(_key, _addr), 711

ChainLog.removeAddress(_key), 711

ChainLog.setAddress(_key, _addr) [X](#) [a](#), 710

definition

ChainLog.location: [mapping](#)(bytes32 => ChainLog.Location), 709

read

ChainLog.get(_index), 711

ChainLog.getAddress(_key), 711

ChainLog.removeAddress(_key), 711

ChainLog.setAddress(_key, _addr) [X](#) [a](#), 710

lock

call

Clipper.file(what, [data](#)) [X](#) [a](#), 243

Clipper.kick(tab, lot, usr, kpr) [X](#) [a](#), 244

Clipper.redo(id, kpr) [X](#), 245
 Clipper.take(id, amt, max, who, [data](#)) [X](#), 246
 Clipper.yank(id) [X](#) [a](#), 249
 D3MHub.exec(ilk) [X](#), 324
 D3MHub.exit(ilk, usr, wad) [X](#), 325
 DssVestMintable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] [X](#) [a](#), 678
 DssVestMintable.file(what, [data](#)) [DssVest] [X](#) [a](#), 677
 DssVestMintable.move(_id, _dst) [DssVest] [X](#), 682
 DssVestMintable.restrict(_id) [DssVest] [X](#), 681
 DssVestMintable.unrestrict(_id) [DssVest] [X](#), 681
 DssVestMintable._vest(_id, _maxAmt) [DssVest], 679
 DssVestMintable._yank(_id, _end) [DssVest], 682
 DssVestSuckable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] [X](#) [a](#), 688
 DssVestSuckable.file(what, [data](#)) [DssVest] [X](#) [a](#), 687
 DssVestSuckable.move(_id, _dst) [DssVest] [X](#), 692
 DssVestSuckable.restrict(_id) [DssVest] [X](#), 690
 DssVestSuckable.unrestrict(_id) [DssVest] [X](#), 691
 DssVestSuckable._vest(_id, _maxAmt) [DssVest], 689
 DssVestSuckable._yank(_id, _end) [DssVest], 691
 DssVestTransferrable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] [X](#) [a](#), 698
 DssVestTransferrable.file(what, [data](#)) [DssVest] [X](#) [a](#), 697
 DssVestTransferrable.move(_id, _dst) [DssVest] [X](#), 702
 DssVestTransferrable.restrict(_id) [DssVest] [X](#), 701
 DssVestTransferrable.unrestrict(_id) [DssVest] [X](#), 701
 DssVestTransferrable._vest(_id, _maxAmt) [DssVest], 699
 DssVestTransferrable._yank(_id, _end) [DssVest], 701

definition

Clipper.[modifier](#) lock(), 242
 D3MHub.[modifier](#) lock(), 320
 DssVestMintable.[modifier](#) lock() [DssVest], 676
 DssVestSuckable.[modifier](#) lock() [DssVest], 685
 DssVestTransferrable.[modifier](#) lock() [DssVest], 695

locked

write

Clipper.[modifier](#) lock(), 242
 ClipperMom.setBreaker(clip, level, delay) [X](#) [a](#), 258
 D3MHub.[modifier](#) lock(), 320
 DssVestMintable.[modifier](#) lock() [DssVest], 676
 DssVestSuckable.[modifier](#) lock() [DssVest], 685
 DssVestTransferrable.[modifier](#) lock() [DssVest], 695

definition

Clipper.locked: [uint256](#), 241
 ClipperMom.locked: [mapping](#)([address](#) => [uint256](#)), 257
 D3MHub.locked: [uint256](#), 319
 DssVestMintable.locked [DssVest] : [uint256](#), 675
 DssVestSuckable.locked [DssVest] : [uint256](#), 685
 DssVestTransferrable.locked [DssVest] : [uint256](#), 695

read

Clipper.[modifier](#) lock(), 242
 ClipperMom.tripBreaker(clip) [X](#), 259
 D3MHub.[modifier](#) lock(), 320
 DssVestMintable.[modifier](#) lock() [DssVest], 676
 DssVestSuckable.[modifier](#) lock() [DssVest], 685
 DssVestTransferrable.[modifier](#) lock() [DssVest], 695

lockETH

call

DssProxyActions.safeLockETH(manager, ethJoin, cdp, owner) [X](#), 534

definition

DssProxyActions.lockETH(manager, ethJoin, cdp) [X](#), 534

lockETHAndDraw

call

DssProxyActions.openLockETHAndDraw(manager, jug, ethJoin, daiJoin, ilk, wadD) [X](#), 538

definition
DssProxyActions.lockETHAndDraw(manager, jug, ethJoin, daiJoin, cdp, wadD) X, 538

lockGem
call
DssProxyActions.safeLockGem(manager, gemJoin, cdp, amt, transferFrom, owner) X, 535
definition
DssProxyActions.lockGem(manager, gemJoin, cdp, amt, transferFrom) X, 535

lockGemAndDraw
call
DssProxyActions.openLockGemAndDraw(manager, jug, gemJoin, daiJoin, ilk, amtC, wadD, transferFrom) X, 539
definition
DssProxyActions.lockGemAndDraw(manager, jug, gemJoin, daiJoin, cdp, amtC, wadD, transferFrom) X, 539

LOG
definition
DssExecLib.LOG: address, 729
read
DssExecLib.getChangelogAddress(_key), 733
DssExecLib.setChangelogAddress(_key, _val) X, 733
DssExecLib.setChangelogIPFS(_ipfsHash) X, 733
DssExecLib.setChangelogSHA256(_SHA256Sum) X, 733
DssExecLib.setChangelogVersion(_version) X, 733

log
definition
DssExec.log: Changelog, 309
read
DssExec.constructor(_expiration, _spellAction) X, 309

logAddress
write
D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341
D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369
D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395
D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415
read
D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341
D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369
D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395
D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415

logIndex
write
D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341
D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369
D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395
D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415
read
D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341
D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369
D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395
D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415

lot
write
Clipper.kick(tab, lot, usr, kpr) X a, 244
Clipper.take(id, amt, max, who, data) X, 246
Flapper.kick(lot, bid) X a, 263
Flopper.dent(id, lot, bid) X, 267
Flopper.kick(gal, lot, bid) X a, 266
Flopper.tick(id) X, 267
read
Clipper.getStatus(id), 248
Clipper.redo(id, kpr) X, 245
Clipper.take(id, amt, max, who, data) X, 246

Clipper.yank(id) [X](#) [a](#), 249
Flapper.deal(id) [X](#), 264
Flapper.tend(id, lot, bid) [X](#), 263
Flopper.deal(id) [X](#), 267
Flopper.dent(id, lot, bid) [X](#), 267
Flopper.tick(id) [X](#), 267

LRC

[definition](#)

[contract](#) LRC, 469

M

make

call

DssProxyActions.makeGemBag(gemJoin) **X**, 534

GemJoin4.make() **X**, 434

definition

DSWethFactory.make() **X**, 513

GemJoin4.make() **X**, 434

GemJoin4.make(usr) **X**, 434

makeAddr

definition

D3MCoreDeployScript.makeAddr(name) [StdCheatsSafe], 343

D3MCoreInitScript.makeAddr(name) [StdCheatsSafe], 371

D3MDeployScript.makeAddr(name) [StdCheatsSafe], 397

D3MInitScript.makeAddr(name) [StdCheatsSafe], 417

makeAddrAndKey

call

D3MCoreDeployScript.makeAddr(name) [StdCheatsSafe], 343

D3MCoreInitScript.makeAddr(name) [StdCheatsSafe], 371

D3MDeployScript.makeAddr(name) [StdCheatsSafe], 397

D3MInitScript.makeAddr(name) [StdCheatsSafe], 417

definition

D3MCoreDeployScript.makeAddrAndKey(name) [StdCheatsSafe], 342

D3MCoreInitScript.makeAddrAndKey(name) [StdCheatsSafe], 370

D3MDeployScript.makeAddrAndKey(name) [StdCheatsSafe], 396

D3MInitScript.makeAddrAndKey(name) [StdCheatsSafe], 417

makeGemBag

call

DssProxyActions.openLockGNTAndDraw(manager, jug, gntJoin, daiJoin, ilk, amtC,
wadD) **X**, 539

definition

DssProxyActions.makeGemBag(gemJoin) **X**, 534

MANA

definition

contract MANA, 471

ManagedGemJoin

definition

contract ManagedGemJoin, 449

mat

write

Spotter.file(ilk, what, **data**) **X** **a**, 228

read

Spotter.poke(ilk) **X**, 228

MATIC

definition

contract MATIC, 473

_max

call

D3MHub._exec(ilk, _pool, Art, lineWad), 322

definition

D3MHub._max(x, y), 320

MAX_BORROW_RATE

definition

D3MCompoundPlan.MAX_BORROW_RATE: **uint256**, 315

read

D3MCompoundPlan.file(what, **data**) **X** **a**, 316

maxDeposit

call

D3MHub._exec(ilk, _pool, Art, lineWad), 322

write

L1DAIBridge.setMaxDeposit(_maxDeposit) **X** **a**, 619

definition

D3MAavePool.maxDeposit(), 351

D3MCompoundPool.maxDeposit(), 379
 L1DAIBridge.maxDeposit: [uint256](#), 617
 read
 L1DAIBridge.deposit(amount, l2Recipient) [X](#), 619
 maximumFee
 write
 USDT.changeFees(_basisPointsRate, _maximumFee) [X](#), 493
 definition
 USDT.maximumFee: [uint256](#), 493
 read
 USDT.transfer(_to, _value) [X](#), 493
 USDT.transferFrom(_from, _to, _value) [X](#), 494
 MAXINT256
 definition
 D3MHub.MAXINT256: [uint256](#), 319
 read
 D3MHub.cull(ilk) [X](#), 326
 D3MHub.exit(ilk, usr, wad) [X](#), 325
 MAX_UINT
 definition
 USDT.MAX_UINT: [uint256](#), 493
 read
 USDT.transferFrom(_from, _to, _value) [X](#), 494
 maxWithdraw
 call
 D3MHub._exec(ilk, _pool, Art, lineWad), 322
 D3MHub._wipe(ilk, _pool, urn), 322
 definition
 D3MAavePool.maxWithdraw(), 351
 D3MCompoundPool.maxWithdraw(), 379
 MCD
 definition
 library MCD, 719
 Median
 definition
 contract Median, 281
 messenger
 write
 L1DAITokenBridge.constructor(_messenger) [OVM_CrossDomainEnabled] [X](#), 592
 L1GovernanceRelay.constructor(_messenger) [OVM_CrossDomainEnabled] [X](#), 600
 L2DAITokenBridge.constructor(_messenger) [OVM_CrossDomainEnabled] [X](#), 604
 L2GovernanceRelay.constructor(_messenger) [OVM_CrossDomainEnabled] [X](#), 610
 definition
 L1DAITokenBridge.messenger [OVM_CrossDomainEnabled] : [address](#), 591
 L1GovernanceRelay.messenger [OVM_CrossDomainEnabled] : [address](#), 599
 L2DAITokenBridge.messenger [OVM_CrossDomainEnabled] : [address](#), 603
 L2GovernanceRelay.messenger [OVM_CrossDomainEnabled] : [address](#), 609
 read
 L1DAITokenBridge.getCrossDomainMessenger() [OVM_CrossDomainEnabled], 592
 L1GovernanceRelay.getCrossDomainMessenger() [OVM_CrossDomainEnabled], 600
 L2DAITokenBridge.getCrossDomainMessenger() [OVM_CrossDomainEnabled], 604
 L2GovernanceRelay.getCrossDomainMessenger() [OVM_CrossDomainEnabled], 610
 L2GovernanceRelay.relay(target, targetData) [X](#), 611
 mgr
 definition
 DssVestMintable.mgr(_id) [DssVest], 676
 DssVestSuckable.mgr(_id) [DssVest], 686
 DssVestTransferrable.mgr(_id) [DssVest], 696
 read
 DssVestMintable.mgr(_id) [DssVest], 676
 DssVestMintable._yank(_id, _end) [DssVest], 682
 DssVestSuckable.mgr(_id) [DssVest], 686
 DssVestSuckable._yank(_id, _end) [DssVest], 691

DssVestTransferrable.mgr(_id) [DssVest], 696
 DssVestTransferrable.yank(_id, _end) [DssVest], 701

MILLION

definition

DssExecLib.MILLION: uint256, 729

min

call

Clipper.take(id, amt, max, who, data) X, 246
 D3MAavePool.maxWithdraw(), 351
 D3MCompoundPool.maxWithdraw(), 379
 D3MHub._exec(ilk, _pool, Art, lineWad), 322
 D3MHub._wipe(ilk, _pool, urn), 322
 Dog.bark(ilk, urn, kpr) X, 253
 DssAutoLine.exec(_ilk) X, 707
 DssVestMintable._vest(_id, _maxAmt) [DssVest], 679
 DssVestSuckable._vest(_id, _maxAmt) [DssVest], 689
 DssVestTransferrable._vest(_id, _maxAmt) [DssVest], 699
 End.skim(ilk, urn) X, 300
 Flopper.dent(id, lot, bid) X, 267
 TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643
 TeleportJoin.settle(sourceDomain, batchedDaiToFlush) X, 645
 Vow.cage() X a, 271

write

ESM.constructor(gem_, end_, proxy_, min_) X, 303
 ESM.file(what, data) X a, 304

definition

Clipper.min(x, y), 243
 D3MAavePool._min(x, y), 350
 D3MCompoundPool._min(x, y), 378
 D3MHub._min(x, y), 320
 Dog.min(x, y), 252
 DssAutoLine.min(x, y), 706
 DssVestMintable.min(x, y) [DssVest], 678
 DssVestSuckable.min(x, y) [DssVest], 687
 DssVestTransferrable.min(x, y) [DssVest], 697
 End.min(x, y), 298
 ESM.min: uint256, 303
 Flopper.min(x, y), 266
 TeleportJoin._min(x, y), 642
 Vow.min(x, y), 270

read

ESM.denyProxy(target) X, 305
 ESM.fire() X, 304

mint

call

D3MCompoundPool.deposit(wad) X, 378
 DaiJoin.exit(usr, wad) X, 426
 DAIMock.constructor() X, 632
 DssVestMintable.pay(_guy, _amt), 683
 Flopper.deal(id) X, 267
 L2DaiGateway.finalizeInboundTransfer(l1Token, from, to, amount,) X, 583
 L2DAITokenBridge.finalizeDeposit(_l1Token, _l2Token, _from, _to, _amount, _data) X, 606
 TeleportJoin.mintPending(teleportGUID, maxFeePercentage, operatorFee) X, 644
 TeleportJoin.requestMint(teleportGUID, maxFeePercentage, operatorFee) X a, 644

definition

Dai.mint(to, value) X a, 590, 616
 Dai.mint(usr, wad) X a, 224
 DAIMock.mint(account, amount) [ERC20], 630
 TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643

mintPending

definition

TeleportJoin.mintPending(teleportGUID, maxFeePercentage, operatorFee) X, 644

mkr

definition

DssExecLib.mkr(), 730

move

call

Clipper.take(id, amt, max, who, **data**) **X**, 246
 D3MHub.wipe(ilk, _pool, urn), 322
 DaiJoin.exit(usr, wad) **X**, 426
 DaiJoin.join(usr, wad) **X**, 426
 DssCdpManager.move(cdp, dst, rad) **X**, 524
 DssProxyActions.draw(manager, jug, daiJoin, cdp, wad) **X**, 536
 DssProxyActions.lockETHAndDraw(manager, jug, ethJoin, daiJoin, cdp, wadD) **X**, 538
 DssProxyActions.lockGemAndDraw(manager, jug, gemJoin, daiJoin, cdp, amtC, wadD, transferFrom) **X**, 539
 DssProxyActions.move(manager, cdp, dst, rad) **X**, 533
 DssPsm.buyGem(usr, gemAmt) **X**, 511
 DssPsm.sellGem(usr, gemAmt) **X**, 510
 End.pack(wad) **X**, 301
 Flapper.cage(rad) **X** **a**, 264
 Flapper.deal(id) **X**, 264
 Flapper.kick(lot, bid) **X** **a**, 263
 Flapper.tend(id, lot, bid) **X**, 263
 Flapper.yank(id) **X**, 264
 Flopper.dent(id, lot, bid) **X**, 267
 Pot.exit(wad) **X**, 279
 Pot.join(wad) **X**, 279
 TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643

definition

Dai.move(src, dst, wad) **X**, 225
 DssCdpManager.move(cdp, dst, rad) **X**, 524
 DssProxyActions.move(manager, cdp, dst, rad) **X**, 533
 DssVestMintable.move(_id, _dst) [DssVest] **X**, 682
 DssVestSuckable.move(_id, _dst) [DssVest] **X**, 692
 DssVestTransferrable.move(_id, _dst) [DssVest] **X**, 702
 Vat.move(src, dst, rad) **X**, 232

_msgData

definition

DAIMock._msgData() [Context], 631

_msgSender

call

DAIMock.approve(spender, amount) [ERC20] **X**, 628
 DAIMock.transfer(to, amount) [ERC20] **X**, 627
 DAIMock.decreaseAllowance(spender, subtractedValue) [ERC20] **X**, 629
 DAIMock.increaseAllowance(spender, addedValue) [ERC20] **X**, 628
 DAIMock.transferFrom(from, to, amount) [ERC20] **X**, 628

definition

DAIMock._msgSender() [Context], 631

mul

call

AuthGemJoin5.exit(usr, amt) **X**, 504
 AuthGemJoin5.join(urn, amt, msgSender) **X** **a**, 504
 AuthGemJoin8.exit(usr, amt) **X**, 506
 AuthGemJoin8.join(urn, amt, msgSender) **X** **a**, 506
 Clipper.getFeedPrice(), 244
 ClipperMom.rmul(x, y), 257
 Clipper.rdiv(x, y), 244
 Clipper.redo(id, kpr) **X**, 245
 Clipper.rmul(x, y), 244
 Clipper.take(id, amt, max, who, **data**) **X**, 246
 Clipper.wmul(x, y), 244
 DaiJoin.exit(usr, wad) **X**, 426
 DaiJoin.join(usr, wad) **X**, 426
 Dog.bark(ilk, urn, kpr) **X**, 253

DssAutoLine.exec(_ilk) [X](#), 707
 DssProxyActions.convertTo18(gemJoin, amt), 530
 DssProxyActionsDsr.exit(daiJoin, pot, wad) [X](#), 548
 DssProxyActionsDsr.exitAll(daiJoin, pot) [X](#), 548
 DssProxyActionsDsr.join(daiJoin, pot, wad) [X](#), 547
 DssProxyActionsEnd.cashETH(ethJoin, end, ilk, wad) [X](#), 544
 DssProxyActionsEnd.cashGem(gemJoin, end, ilk, wad) [X](#), 545
 DssProxyActions._getDrawDart(vat, jug, urn, ilk, wad), 530
 DssProxyActions._getWipeAllWad(vat, usr, urn, ilk), 530
 DssProxyActions.toRad(wad), 529
 DssPsm.buyGem(usr, gemAmt) [X](#), 511
 DssPsm.sellGem(usr, gemAmt) [X](#), 510
 DssVestMintable.accrued(_time, _bgn, _fin, _tot) [DssVest], 680
 DssVestSuckable.accrued(_time, _bgn, _fin, _tot) [DssVest], 690
 DssVestSuckable.pay(_guy, _amt), 693
 DssVestTransferrable.accrued(_time, _bgn, _fin, _tot) [DssVest], 700
 End.flow(ilk) [X](#), 300
 End.pack(wad) [X](#), 301
 End.rmul(x, y), 298
 End.wdiv(x, y), 298
 Flapper.tend(id, lot, bid) [X](#), 263
 Flopper.dent(id, lot, bid) [X](#), 267
 Flopper.tick(id) [X](#), 267
 GemJoin3.exit(usr, amt) [X](#), 430
 GemJoin3.join(usr, amt) [X](#), 430
 GemJoin5.exit(usr, amt) [X](#), 436
 GemJoin5.join(usr, amt) [X](#), 436
 GemJoin7.exit(usr, amt) [X](#), 440
 GemJoin7.join(usr, amt) [X](#), 440
 GemJoin8.exit(usr, amt) [X](#), 442
 GemJoin8.join(usr, amt) [X](#), 442
 LinearDecrease.price(top, dur), 236
 ManagedGemJoin.exit(urn, usr, amt) [X](#) [a](#), 450
 ManagedGemJoin.join(usr, amt) [X](#) [a](#), 450
 PAXG.getFeeFor(_value), 479
 Pot.drip() [X](#), 279
 Pot.exit(wad) [X](#), 279
 Pot.join(wad) [X](#), 279
 Pot.rmul(x, y), 278
 Spotter.poke(ilk) [X](#), 228
 Spotter.rdiv(x, y), 228
 USDT.transfer(_to, _value) [X](#), 493
 USDT.transferFrom(_from, _to, _value) [X](#), 494
 Vat.fold(i, u, rate) [X](#) [a](#), 234
 Vat.fork(ilk, src, dst, dink, dart) [X](#), 233
 Vat.frob(i, u, v, w, dink, dart) [X](#), 232
 Vat.grab(i, u, v, w, dink, dart) [X](#) [a](#), 233

definition

AuthGemJoin5.mul(x, y), 504
 AuthGemJoin8.mul(x, y), 506
 ClipperMom.mul(x, y), 257
 Clipper.mul(x, y), 244
 DaiJoin.mul(x, y), 426
 Dog.mul(x, y), 252
 DssAutoLine.mul(x, y), 706
 DssProxyActionsDsr.mul(x, y) [Common], 547
 DssProxyActionsEnd.mul(x, y) [Common], 543
 DssProxyActions.mul(x, y) [Common], 529
 DssPsm.mul(x, y), 510
 DssVestMintable.mul(x, y) [DssVest], 678
 DssVestSuckable.mul(x, y) [DssVest], 688
 DssVestTransferrable.mul(x, y) [DssVest], 698
 End.mul(x, y), 298

Flapper.mul(x, y), 262
Flopper.mul(x, y), 266
GemJoin2.mul(x, y), 428
GemJoin3.mul(x, y), 430
GemJoin5.mul(x, y), 436
GemJoin7.mul(x, y), 440
GemJoin8.mul(x, y), 442
LinearDecrease.mul(x, y), 236
ManagedGemJoin.mul(x, y), 450
PAXG.mul(a, b), 477
Pot.mul(x, y), 278
SafeMath.mul(a, b), 753
Spotter.mul(x, y), 228
Vat.mul(x, y), 230, 231

MULTICALL3_ADDRESS

definition

D3MCoreDeployScript.MULTICALL3_ADDRESS [CommonBase] : [address](#), 329
D3MCoreInitScript.MULTICALL3_ADDRESS [CommonBase] : [address](#), 357
D3MDeployScript.MULTICALL3_ADDRESS [CommonBase] : [address](#), 383
D3MInitScript.MULTICALL3_ADDRESS [CommonBase] : [address](#), 403

multiplierPerBlock

call

D3MCompoundPlan.calculateTargetSupply(targetInterestRate, borrows), 316

N

NAME

definition

D3MCoreDeployScript.NAME: [string](#), 329

D3MCoreInitScript.NAME: [string](#), 357

read

D3MCoreDeployScript.run() [X](#), 346

D3MCoreInitScript.run() [X](#), 374

name

call

D3MInit._init(dss, d3m, cfg, gem), 725

GemInfo.name(token), 665

IlkRegistry.add(adapter) [X](#), 668

write

DAIMock.constructor(name_, symbol_) [ERC20] [X](#), 626

IlkRegistry.file(ilk, what, data) [X](#) [a](#), 670

definition

AAVE.name: [string](#), 451

BAL.name: [string](#), 453

BAT.name: [string](#), 455

COMP.name: [string](#), 457

DAIMock.name [ERC20] : [string](#), 625

DAIMock.name() [ERC20], 626

Dai.name: [string](#), 223, 587, 613

DGD.name: [string](#), 459

GemInfo.name(token), 665

GNT.name: [string](#), 461

GUSD.name: [string](#), 463

IlkRegistry.name(ilk), 672

KNC.name: [string](#), 465

LINK.name: [string](#), 467

LRC.name: [string](#), 469

MANA.name: [string](#), 471

MATIC.name: [string](#), 473

OMG.name: [string](#), 475

PAXG.name: [string](#), 477

PAXUSD.name: [string](#), 481

RENBTC.name: [string](#), 483

REP.name: [string](#), 485

TUSD.name: [string](#), 487

UNI.name: [string](#), 489

USDC.name: [string](#), 491

USDT.name: [string](#), 493

WBTC.name: [string](#), 495

WETH9.name: [string](#), 517

WSTETH.name: [string](#), 497

YFI.name: [string](#), 499

ZRX.name: [string](#), 501

read

Dai.constructor(chainId_) [X](#), 224

Dai._calculateDomainSeparator(chainId), 588, 614

DAIMock.name() [ERC20], 626

IlkRegistry.name(ilk), 672

newIlkLerp

call

DssExecLib.linearInterpolation(_name, _target, _ilk, _what, _startTime, _start, _end, _duration) [X](#), 752

newLerp

call

DssExecLib.linearInterpolation(_name, _target, _what, _startTime, _start, _end, _duration) [X](#), 751

newUser

definition


```

    MCD.newUser(dss), 721
next
  write
    DssCdpManager.give(cdp, dst) X, 523
    DssCdpManager.open(ilk, usr) X, 522
  read
    DssCdpManager.give(cdp, dst) X, 523
nextCastTime
  call
    DssAction.nextCastTime(eta), 308
    DssExec.nextCastTime(), 309
  definition
    DssAction.nextCastTime(eta), 308
    DssExecLib.nextCastTime(_eta, _ts, _officeHours), 735
    DssExec.nextCastTime(), 309
noGasMetering
  definition
    D3MCoreDeployScript.modifier noGasMetering() [StdCheatsSafe], 335
    D3MCoreInitScript.modifier noGasMetering() [StdCheatsSafe], 363
    D3MDeployScript.modifier noGasMetering() [StdCheatsSafe], 389
    D3MInitScript.modifier noGasMetering() [StdCheatsSafe], 409
nonces
  write
    Dai.permit(holder, spender, nonce, expiry, allowed, v, r, s) X, 225
    Dai.permit(owner, spender, value, deadline, v, r, s) X, 590, 616
  definition
    Dai.nonces: mapping(address => uint256), 223, 587, 613
  read
    Dai.permit(holder, spender, nonce, expiry, allowed, v, r, s) X, 225
    Dai.permit(owner, spender, value, deadline, v, r, s) X, 590, 616
nope
  call
    D3MAavePool.file(what, data) X a, 350
    D3MCompoundPool.file(what, data) X a, 378
    DssExecLib.undelegateVat(_usr) X, 734
    DssProxyActions.nope(obj, usr) X, 532
    DssPsm.nope(usr) X a, 510
    Vow.file(what, data) X a, 270
  definition
    DssProxyActions.nope(obj, usr) X, 532
    DssPsm.nope(usr) X a, 510
    Vat.nope(usr) X, 230
note
  call
    Median.deny(usr) X a, 282
    Median.diss(a) X a, 284
    Median.drop(a) X a, 283
    Median.kiss(a) X a, 284
    Median.lift(a) X a, 283
    Median.rely(usr) X a, 282
    Median.setBar(bar_) X a, 284
    OSM.change(src_) X a, 287
    OSM.deny(usr) X a, 286
    OSM.diss(a) X a, 288
    OSM.kiss(a) X a, 288
    OSM.poke() X, 287
    OSM.rely(usr) X a, 286
    OSM.start() X a, 286
    OSM.stop() X a, 286
    OSM.void() X a, 287
  definition
    Median.modifier note() [LibNote], 281
    OSM.modifier note() [LibNote(2)], 285

```

numDomains
 definition
 TeleportRouter.numDomains(), 654

nxt
 write
 OSM.poke() **X**, 287
 OSM.void() **X** **a**, 287
 definition
 OSM.nxt: OSM.Feed, 285

read
 OSM.peep(), 288
 OSM.poke() **X**, 287
 OSM.void() **X** **a**, 287

O

officeHours

call

DssAction.modifier limited(), 307
 DssAction.nextCastTime(eta), 308
 DssExec.officeHours(), 309

definition

DssAction.officeHours(), 307
 DssExec.officeHours(), 309

offset

definition

L2DaiGateway.offset [L2CrossDomainEnabled] : uint160, 581
 L2GovernanceRelay.offset [L2CrossDomainEnabled] : uint160, 585

read

L2DaiGateway.applyL1ToL2Alias(l1Address) [L2CrossDomainEnabled], 581
 L2GovernanceRelay.applyL1ToL2Alias(l1Address) [L2CrossDomainEnabled], 585

OMG

definition

contract OMG, 475

ONE

definition

DaiJoin.ONE: uint256, 425
 Flapper.ONE: uint256, 261
 Flopper.ONE: uint256, 265
 Jug.ONE: uint256, 273
 Pot.ONE: uint256, 277
 Spotter.ONE: uint256, 227

read

DaiJoin.exit(usr, wad) X, 426
 DaiJoin.join(usr, wad) X, 426
 Flapper.tend(id, lot, bid) X, 263
 Flopper.dend(id, lot, bid) X, 267
 Flopper.tick(id) X, 267
 Jug.drip(ilk) X, 275
 Jug.init(ilk) X a, 274
 Jug._rmul(x, y), 274
 Pot.constructor(vat_) X, 277
 Pot.cage() X a, 279
 Pot.drip() X, 279
 Pot._rmul(x, y), 278
 Spotter.constructor(vat_) X, 227
 Spotter.rdiv(x, y), 228

ONE_HOUR

definition

OSM.ONE_HOUR: uint16, 285

onlyFromCrossDomainAccount

call

L1DAITokenBridge.finalizeERC20Withdrawal(_l1Token, _l2Token, _from, _to, _amount, _data) X, 595
 L2DAITokenBridge.finalizeDeposit(_l1Token, _l2Token, _from, _to, _amount, _data) X, 606
 L2GovernanceRelay.relay(target, targetData) X, 611

definition

L1DAITokenBridge.modifier onlyFromCrossDomainAccount(_sourceDomainAccount) [OVM_CrossDomainEnabled], 592
 L1GovernanceRelay.modifier onlyFromCrossDomainAccount(_sourceDomainAccount) [OVM_CrossDomainEnabled], 599
 L2DAITokenBridge.modifier onlyFromCrossDomainAccount(_sourceDomainAccount) [OVM_CrossDomainEnabled], 603
 L2GovernanceRelay.modifier onlyFromCrossDomainAccount(_sourceDomainAccount) [OVM_CrossDomainEnabled], 609

onlyHub

call

D3MAavePool.deposit(wad) [X](#), 350
 D3MAavePool.exit(dst, wad) [X](#), 351
 D3MAavePool.withdraw(wad) [X](#), 351
 D3MCompoundPool.deposit(wad) [X](#), 378
 D3MCompoundPool.exit(dst, wad) [X](#), 379
 D3MCompoundPool.withdraw(wad) [X](#), 379
 definition
 D3MAavePool.modifier onlyHub(), 349
 D3MCompoundPool.modifier onlyHub(), 377
 onlyL1Counterpart
 call
 L2DaiGateway.finalizeInboundTransfer(l1Token, from, to, amount,) [X](#), 583
 L2GovernanceRelay.relay(target, targetData) [X](#), 586
 definition
 L2DaiGateway.modifier onlyL1Counterpart(l1Counterpart) [L2CrossDomainEnabled], 581
 L2GovernanceRelay.modifier onlyL1Counterpart(l1Counterpart)
 [L2CrossDomainEnabled], 585
 onlyL2Counterpart
 call
 L1DaiGateway.finalizeInboundTransfer(l1Token, from, to, amount, data) [X](#), 572
 definition
 L1DaiGateway.modifier onlyL2Counterpart(l2Counterpart) [L1CrossDomainEnabled], 569
 L1GovernanceRelay.modifier onlyL2Counterpart(l2Counterpart)
 [L1CrossDomainEnabled], 577
 onlyOwner
 call
 ClipperMom.setAuthority(authority_) [X](#), 258
 ClipperMom.setOwner(owner_) [X](#), 258
 ClipperMom.setPriceTolerance(clip, value) [X](#), 258
 D3MMom.setAuthority(authority_) [X](#), 356
 D3MMom.setOwner(owner_) [X](#), 355
 L1EscrowMom.setAuthority(authority_) [X](#), 638
 L1EscrowMom.setOwner(owner_) [X](#), 637
 definition
 ClipperMom.modifier onlyOwner(), 257
 D3MMom.modifier onlyOwner(), 355
 L1EscrowMom.modifier onlyOwner(), 637
 onlyPayloadSize
 call
 USDT.approve(_spender, _value) [X](#), 494
 USDT.transfer(_to, _value) [X](#), 493
 USDT.transferFrom(_from, _to, _value) [X](#), 494
 definition
 USDT.modifier onlyPayloadSize(size), 493
 open
 call
 DssProxyActions.open(manager, ilk, usr) [X](#), 532
 DssProxyActions.openLockETHAndDraw(manager, jug, ethJoin, daiJoin, ilk, wadD) [X](#),
 538
 DssProxyActions.openLockGemAndDraw(manager, jug, gemJoin, daiJoin, ilk, amtC,
 wadD, transferFrom) [X](#), 539
 definition
 DssCdpManager.open(ilk, usr) [X](#), 522
 DssProxyActions.open(manager, ilk, usr) [X](#), 532
 openLockETHAndDraw
 definition
 DssProxyActions.openLockETHAndDraw(manager, jug, ethJoin, daiJoin, ilk, wadD) [X](#),
 538
 openLockGemAndDraw
 call
 DssProxyActions.openLockGNTAndDraw(manager, jug, gntJoin, daiJoin, ilk, amtC,
 wadD) [X](#), 539
 definition

```

        DssProxyActions.openLockGemAndDraw(manager, jug, gemJoin, daiJoin, ilk, amtC,
            wadD, transferFrom) X, 539
openLockGNTAndDraw
    definition
        DssProxyActions.openLockGNTAndDraw(manager, jug, gntJoin, daiJoin, ilk, amtC,
            wadD) X, 539
OPTIMAL_UTILIZATION_RATE
    call
        D3MAavePlan._calculateTargetSupply(targetInterestRate, totalDebt), 313
OptionalLoadDependencies
    definition
        contract OptionalLoadDependencies, 381
oracleAuth
    write
        BasicRelay.constructor(_oracleAuth, _daiJoin) X, 657
        TrustedRelay.constructor(_oracleAuth, _daiJoin, _ethPriceOracle) X, 661
    definition
        BasicRelay.oracleAuth: TeleportOracleAuthLike, 657
        TrustedRelay.oracleAuth: TeleportOracleAuthLike, 661
    read
        BasicRelay.constructor(_oracleAuth, _daiJoin) X, 657
        BasicRelay.relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v,
            r, s) X, 658
        TrustedRelay.constructor(_oracleAuth, _daiJoin, _ethPriceOracle) X, 661
        TrustedRelay.requestMint(teleportGUID, signatures, maxFeePercentage, gasFee,
            expiry, v, r, s), 664
orb1
    call
        DssExecLib.whitelistOracleMedians(_oracle) X, 747
orcl
    write
        Median.drop(a) X a, 283
        Median.lift(a) X a, 283
    definition
        Median.orcl: mapping(address => uint256), 281
    read
        Median.poke(val_, age_, v, r, s) X, 283
OSM
    definition
        contract OSM, 285
osmMom
    call
        DssExecLib.addNewCollateral(co) X, 750
        DssExecLib.allowOSMFreeze(_osm, _ilk) X, 748
    definition
        DssExecLib.osmMom(), 731
otc
    write
        CalleeMakerOtcDai.setUp(otc_, daiJoin_) [CalleeMakerOtc], 549
    definition
        CalleeMakerOtcDai.otc [CalleeMakerOtc] : OtcLike, 549
    read
        CalleeMakerOtcDai.clipperCall(sender, daiAmt, gemAmt, data) X, 550
out
    write
        End.cash(ilk, wad) X, 301
    definition
        End.out: mapping(bytes32 => mapping(address => uint256)), 295
    read
        End.cash(ilk, wad) X, 301
outboundTransfer
    call
        L2DaiGateway.outboundTransfer(l1Token, to, amount, data) X, 582

```

definition

L1DaiGateway.outboundTransfer(l1Token, to, amount, maxGas, gasPriceBid, data) X, 571
 L2DaiGateway.outboundTransfer(l1Token, to, amount, , , data) X, 583
 L2DaiGateway.outboundTransfer(l1Token, to, amount, data) X, 582

owner

call

DssProxyActions.giveToProxy(proxyRegistry, manager, cdp, dst) X, 532

write

ClipperMom.constructor(spotter_) X, 257
 ClipperMom.setOwner(owner_) X, 258
 D3MMom.constructor() X, 355
 D3MMom.setOwner(owner_) X, 355
 L1EscrowMom.constructor(escrow_, token_) X, 637
 L1EscrowMom.setOwner(owner_) X, 637

definition

ClipperMom.owner: address, 257
 D3MMom.owner: address, 355
 L1EscrowMom.owner: address, 637
 USDT.owner: address, 493

read

ClipperMom.modifier onlyOwner(), 257
 ClipperMom.isAuthorized(src, sig), 258
 ClipperMom.setOwner(owner_) X, 258
 D3MMom.modifier onlyOwner(), 355
 D3MMom.isAuthorized(src, sig), 355
 L1EscrowMom.modifier onlyOwner(), 637
 L1EscrowMom.isAuthorized(src, sig), 637
 L1EscrowMom.setOwner(owner_) X, 637
 USDT.transfer(_to, _value) X, 493
 USDT.transferFrom(_from, _to, _value) X, 494

owns

call

DssProxyActions.safeLockETH(manager, ethJoin, cdp, owner) X, 534
 DssProxyActions.safeLockGem(manager, gemJoin, cdp, amt, transferFrom, owner) X, 535
 DssProxyActions.safeWipe(manager, daiJoin, cdp, wad, owner) X, 537
 DssProxyActions.safeWipeAll(manager, daiJoin, cdp, owner) X, 538
 DssProxyActions.wipe(manager, daiJoin, cdp, wad) X, 537
 DssProxyActions.wipeAll(manager, daiJoin, cdp) X, 537

write

DssCdpManager.give(cdp, dst) X, 523
 DssCdpManager.open(ilk, usr) X, 522

definition

DssCdpManager.owns: mapping(uint256 => address), 521

read

DssCdpManager.modifier cdpAllowed(cdp), 521
 DssCdpManager.cdpAllow(cdp, usr, ok) X, 522
 DssCdpManager.give(cdp, dst) X, 523

P

pack

call

DssProxyActionsEnd.pack(daiJoin, end, wad) **X**, 544

definition

DssProxyActionsEnd.pack(daiJoin, end, wad) **X**, 544

End.pack(wad) **X**, 301

pad

write

Flopper.file(what, **data**) **X** **a**, 266

definition

Flopper.pad: **uint256**, 265

read

Flopper.tick(id) **X**, 267

par

call

Clipper.getFeedPrice(), 244

End.cage(ilk) **X**, 299

write

Spotter.constructor(vat_) **X**, 227

Spotter.file(what, **data**) **X** **a**, 228

definition

Spotter.par: **uint256**, 227

read

Spotter.poke(ilk) **X**, 228

parseJson

call

D3MCoreDeployScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 339

D3MCoreDeployScript.readReceipt(path, index) [StdCheatsSafe], 340

D3MCoreDeployScript.readReceipts(path) [StdCheatsSafe], 340

D3MCoreDeployScript.readTx1559(path, index) [StdCheatsSafe], 340

D3MCoreDeployScript.readTx1559s(path) [StdCheatsSafe], 340

D3MCoreInitScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 367

D3MCoreInitScript.readReceipt(path, index) [StdCheatsSafe], 368

D3MCoreInitScript.readReceipts(path) [StdCheatsSafe], 368

D3MCoreInitScript.readTx1559(path, index) [StdCheatsSafe], 368

D3MCoreInitScript.readTx1559s(path) [StdCheatsSafe], 368

D3MDeployScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 393

D3MDeployScript.readReceipt(path, index) [StdCheatsSafe], 394

D3MDeployScript.readReceipts(path) [StdCheatsSafe], 394

D3MDeployScript.readTx1559(path, index) [StdCheatsSafe], 394

D3MDeployScript.readTx1559s(path) [StdCheatsSafe], 394

D3MInitScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 413

D3MInitScript.readReceipt(path, index) [StdCheatsSafe], 414

D3MInitScript.readReceipts(path) [StdCheatsSafe], 414

D3MInitScript.readTx1559(path, index) [StdCheatsSafe], 414

D3MInitScript.readTx1559s(path) [StdCheatsSafe], 414

parseOutboundData

call

L1DaiGateway.outboundTransfer(l1Token, to, amount, maxGas, gasPriceBid, **data**) **X**, 571

L2DaiGateway.outboundTransfer(l1Token, to, amount, , , **data**) **X**, 583

definition

L1DaiGateway.parseOutboundData(**data**), 572

L2DaiGateway.parseOutboundData(**data**), 584

pass

call

OSM.poke() **X**, 287

definition

OSM.pass(), 287

pause

write

DssExec.constructor(_expiration, _spellAction) **X**, 309

definition
 DssExec.pause: PauseAbstract, 309
 read
 DssExec.cast() **X**, 310
 DssExec.schedule() **X**, 309
 pauseGasMetering
 call
 D3MCoreDeployScript.modifier noGasMetering() [StdCheatsSafe], 335
 D3MCoreInitScript.modifier noGasMetering() [StdCheatsSafe], 363
 D3MDeployScript.modifier noGasMetering() [StdCheatsSafe], 389
 D3MInitScript.modifier noGasMetering() [StdCheatsSafe], 409
 pauseProxy
 definition
 DssExecLib.pauseProxy(), 732
 PAXG
 definition
 contract PAXG, 477
 PAXUSD
 definition
 contract PAXUSD, 481
 pay
 call
 DssVestMintable._vest(_id, _maxAmt) [DssVest], 679
 DssVestSuckable._vest(_id, _maxAmt) [DssVest], 689
 DssVestTransferrable._vest(_id, _maxAmt) [DssVest], 699
 definition
 DssVestMintable.pay(_guy, _amt), 683
 DssVestSuckable.pay(_guy, _amt), 693
 DssVestTransferrable.pay(_guy, _amt), 703
 peek
 call
 Clipper.getFeedPrice(), 244
 ClipperMom.getPrices(clip), 258
 D3MOracle.read(), 354
 OSM.poke() **X**, 287
 Spotter.poke(ilk) **X**, 228
 TrustedRelay.relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s) **X**, 663
 definition
 D3MOracle.peek(), 354
 Median.peek(), 282
 OSM.peek(), 287
 Value.peek(), 289
 peep
 call
 ClipperMom.getPrices(clip), 258
 definition
 OSM.peep(), 288
 pending
 write
 TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643
 TeleportJoin.requestMint(teleportGUID, maxFeePercentage, operatorFee) **X** **a**, 644
 read
 TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643
 permit
 definition
 Dai.permit(holder, spender, nonce, expiry, allowed, v, r, s) **X**, 225
 Dai.permit(owner, spender, **value**, deadline, v, r, s) **X**, 590, 616
 PERMIT_TYPEHASH
 definition
 Dai.PERMIT_TYPEHASH: **bytes32**, 223, 587, 613
 read
 Dai.permit(holder, spender, nonce, expiry, allowed, v, r, s) **X**, 225


```

    Dai.permit(owner, spender, value, deadline, v, r, s) X, 590, 616
Pie
  write
    Pot.exit(wad) X, 279
    Pot.join(wad) X, 279
  definition
    Pot.Pie: uint256, 277
  read
    Pot.drip() X, 279
    Pot.exit(wad) X, 279
    Pot.join(wad) X, 279
pie
  call
    DssProxyActionsDsr.exitAll(daiJoin, pot) X, 548
  write
    Pot.exit(wad) X, 279
    Pot.join(wad) X, 279
  definition
    Pot.pie: mapping(address => uint256), 277
  read
    Pot.exit(wad) X, 279
    Pot.join(wad) X, 279
pip
  write
    IlkRegistry.file(ilk, what, data) X a, 669
    IlkRegistry.update(ilk) X, 673
    Spotter.file(ilk, what, pip-) X a, 228
  definition
    IlkRegistry.pip(ilk), 672
  read
    IlkRegistry.pip(ilk), 672
    Spotter.poke(ilk) X, 228
plan
  write
    D3MHub.file(ilk, what, data) X a, 321
  definition
    D3MHub.plan(ilk), 327
  read
    D3MHub.exec(ilk, _pool, Art, lineWad), 322
    D3MHub.plan(ilk), 327
plot
  call
    DssExec.schedule() X, 309
poke
  call
    D3MInit.init(dss, d3m, cfg, gem), 725
    DssExecLib.updateCollateralPrice(_ilk) X, 736
    MCD.initIlk(dss, ilk), 720
    MCD.initIlk(dss, ilk, join), 720
    MCD.initIlk(dss, ilk, join, pip), 720
  definition
    Median.poke(val_, age_, v, r, s) X, 283
    OSM.poke() X, 287
    Spotter.poke(ilk) X, 228
pool
  write
    D3MAavePlan.constructor(dai_, pool_) X, 311
    D3MAavePool.constructor(ilk_, hub_, dai_, pool_) X, 349
    D3MHub.file(ilk, what, data) X a, 321
  definition
    D3MAavePlan.pool: LendingPoolLike, 311
    D3MAavePool.pool: LendingPoolLike, 349
    D3MHub.pool(ilk), 327

```

read

D3MAavePlan.active(), 314
 D3MAavePlan.constructor(dai_, pool_) X, 311
 D3MAavePool.constructor(ilk_, hub_, dai_, pool_) X, 349
 D3MAavePool.deposit(wad) X, 350
 D3MAavePool.withdraw(wad) X, 351
 D3MHub.cull(ilk) X, 326
 D3MHub.exec(ilk) X, 324
 D3MHub.exit(ilk, usr, wad) X, 325
 D3MHub.pool(ilk), 327
 D3MHub.uncull(ilk) X, 326

pos

write

ChainLog.removeAddress(_key), 711
 Clipper.kick(tab, lot, usr, kpr) X a, 244
 Clipper.remove(id), 247
 Cure.drop(src) X a, 293
 Cure.lift(src) X a, 293
 IlkRegistry.remove(ilk), 670

definition

Cure.pos: mapping(address => uint256), 291
 IlkRegistry.pos(ilk), 671

read

ChainLog.removeAddress(_key), 711
 ChainLog.setAddress(_key, _addr) X a, 710
 Clipper.remove(id), 247
 Cure.drop(src) X a, 293
 Cure.lift(src) X a, 293
 Cure.load(src) X, 293
 IlkRegistry.pos(ilk), 671
 IlkRegistry.put(_ilk, _join, _gem, _dec, _class, _pip, _xlip, _name, _symbol) X a, 673
 IlkRegistry.remove(ilk), 670

postDebtChange

call

D3MHub.exec(ilk) X, 324

Pot

definition

contract Pot, 277

pot

call

DssExecLib.accumulatedDSR() X, 735
 DssExecLib.setDSR(_rate, _doDrip) X, 737

write

End.file(what, data) X a, 298

definition

DssExecLib.pot(), 731
 End.pot: PotLike, 295

read

End.cage() X a, 299

preDebtChange

call

D3MHub.exec(ilk) X, 324

definition

D3MCompoundPool.preDebtChange() X, 379

prev

call

OSM.poke() X, 287

write

DssCdpManager.give(cdp, dst) X, 523
 DssCdpManager.open(ilk, usr) X, 522

definition

OSM.prev(ts), 287

```

    read
      DssCdpManager.give(cdp, dst) X, 523
price
  call
    Clipper.status(tic, top), 248
  definition
    ExponentialDecrease.price(top, dur), 240
    LinearDecrease.price(top, dur), 236
    StairstepExponentialDecrease.price(top, dur), 238
projectRoot
  call
    ScriptTools.exportContract(name, label, addr), 715
    ScriptTools.readInput(name), 713
    ScriptTools.readOutput(name), 714
    ScriptTools.readOutput(name, timestamp), 713
proxies
  call
    DssProxyActions.giveToProxy(proxyRegistry, manager, cdp, dst) X, 532
proxy
  write
    ESM.constructor(gem_, end_, proxy_, min_) X, 303
  definition
    ESM.proxy: address, 303
  read
    ESM.denyProxy(target) X, 305
    ESM.fire() X, 304
    ESM.revokesGovernanceAccess(), 303
PSMCallee
  definition
    contract PSMCallee, 555
pull
  definition
    Dai.pull(usr, wad) X, 225
push
  definition
    Dai.push(usr, wad) X, 225
put
  call
    D3MInit._init(dss, d3m, cfg, gem), 725
  definition
    IlkRegistry.put(_ilk, _join, _gem, _dec, _class, _pip, _xlip, _name, _symbol) X a,
    673

```

Q

quit

call

DssProxyActionsEnd._free(manager, end, cdp), 543

DssProxyActions.quit(manager, cdp, dst) **X**, 533

definition

D3MAavePool.quit(dst) **X** **a**, 351

D3MCompoundPool.quit(dst) **X** **a**, 379

DssCdpManager.quit(cdp, dst) **X**, 524

DssProxyActions.quit(manager, cdp, dst) **X**, 533

R**RAD****definition**

D3MInitScript.RAD: [uint256](#), 403

DssExecLib.RAD: [uint256](#), 729

MCD.RAD: [uint256](#), 719

read

D3MInitScript.run() [X](#), 420

DssExecLib.decreaseGlobalDebtCeiling(_amount) [X](#), 737

DssExecLib.decreaseIlkDebtCeiling(_ilk, _amount, _global) [X](#), 742

DssExecLib.increaseGlobalDebtCeiling(_amount) [X](#), 737

DssExecLib.increaseIlkDebtCeiling(_ilk, _amount, _global) [X](#), 741

DssExecLib.sendPaymentFromSurplusBuffer(_target, _amount) [X](#), 751

DssExecLib.setDebtAuctionDAIAmount(_amount) [X](#), 739

DssExecLib.setGlobalDebtCeiling(_amount) [X](#), 737

DssExecLib.setIlkAutoLineDebtCeiling(_ilk, _amount) [X](#), 743

DssExecLib.setIlkAutoLineParameters(_ilk, _amount, _gap, _ttl) [X](#), 742

DssExecLib.setIlkDebtCeiling(_ilk, _amount) [X](#), 741

DssExecLib.setIlkMaxLiquidationAmount(_ilk, _amount) [X](#), 744

DssExecLib.setIlkMinVaultAmount(_ilk, _amount) [X](#), 743

DssExecLib.setKeeperIncentiveFlatRate(_ilk, _amount) [X](#), 745

DssExecLib.setMaxTotalDAILiquidationAmount(_amount) [X](#), 740

DssExecLib.setMaxTotalDAILiquidationAmountLEGACY(_amount) [X](#), 740

DssExecLib.setSurplusAuctionAmount(_amount) [X](#), 738

DssExecLib.setSurplusBuffer(_amount) [X](#), 738

rate**write**

Vat.init(ilk) [X](#) [a](#), 231

read

Vat.init(ilk) [X](#) [a](#), 231

RATES.ONE_HUNDRED_PCT**definition**

DssExecLib.RATES.ONE_HUNDRED_PCT: [uint256](#), 729

read

DssExecLib.setDSR(_rate, _doDrip) [X](#), 737

DssExecLib.setGlobalStabilityFee(_rate) [X](#), 741

DssExecLib.setIlkStabilityFee(_ilk, _rate, _doDrip) [X](#), 746

RawEIP1559ScriptArtifact**definition**

struct StdCheatsSafe.RawEIP1559ScriptArtifact, 334, 362, 388, 408

RawReceipt**definition**

struct StdCheatsSafe.RawReceipt, 333, 361, 387, 407

RawReceiptLog**definition**

struct StdCheatsSafe.RawReceiptLog, 334, 362, 388, 408

rawToConvertedEIP1559Detail**call**

D3MCoreDeployScript.rawToConvertedEIP1559Detail(rawTx) [StdCheatsSafe], 339

D3MCoreInitScript.rawToConvertedEIP1559Detail(rawTx) [StdCheatsSafe], 367

D3MDeployScript.rawToConvertedEIP1559Detail(rawTx) [StdCheatsSafe], 393

D3MInitScript.rawToConvertedEIP1559Detail(rawTx) [StdCheatsSafe], 413

definition

D3MCoreDeployScript.rawToConvertedEIP1559DetailDetail(rawDetail) [StdCheatsSafe], 339

D3MCoreInitScript.rawToConvertedEIP1559DetailDetail(rawDetail) [StdCheatsSafe], 367

D3MDeployScript.rawToConvertedEIP1559DetailDetail(rawDetail) [StdCheatsSafe], 393

D3MInitScript.rawToConvertedEIP1559DetailDetail(rawDetail) [StdCheatsSafe], 414

rawToConvertedEIP1559**call**

D3MCoreDeployScript.rawToConvertedEIP1559s(rawTx) [StdCheatsSafe], 339

D3MCoreDeployScript.readTx1559(path, index) [StdCheatsSafe], 340

D3MCoreInitScript.rawToConvertedEIP1559s(rawTx) [StdCheatsSafe], 367

D3MCoreInitScript.readTx1559(path, index) [StdCheatsSafe], 368

D3MDeployScript.rawToConvertedEIPTx1559s(rawTxs) [StdCheatsSafe], 393
 D3MDeployScript.readTx1559(path, index) [StdCheatsSafe], 394
 D3MInitScript.rawToConvertedEIPTx1559s(rawTxs) [StdCheatsSafe], 413
 D3MInitScript.readTx1559(path, index) [StdCheatsSafe], 414

definition

D3MCoreDeployScript.rawToConvertedEIPTx1559(rawTx) [StdCheatsSafe], 339
 D3MCoreInitScript.rawToConvertedEIPTx1559(rawTx) [StdCheatsSafe], 367
 D3MDeployScript.rawToConvertedEIPTx1559(rawTx) [StdCheatsSafe], 393
 D3MInitScript.rawToConvertedEIPTx1559(rawTx) [StdCheatsSafe], 413

rawToConvertedEIPTx1559s

call

D3MCoreDeployScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 339
 D3MCoreDeployScript.readTx1559s(path) [StdCheatsSafe], 340
 D3MCoreInitScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 367
 D3MCoreInitScript.readTx1559s(path) [StdCheatsSafe], 368
 D3MDeployScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 393
 D3MDeployScript.readTx1559s(path) [StdCheatsSafe], 394
 D3MInitScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 413
 D3MInitScript.readTx1559s(path) [StdCheatsSafe], 414

definition

D3MCoreDeployScript.rawToConvertedEIPTx1559s(rawTxs) [StdCheatsSafe], 339
 D3MCoreInitScript.rawToConvertedEIPTx1559s(rawTxs) [StdCheatsSafe], 367
 D3MDeployScript.rawToConvertedEIPTx1559s(rawTxs) [StdCheatsSafe], 393
 D3MInitScript.rawToConvertedEIPTx1559s(rawTxs) [StdCheatsSafe], 413

rawToConvertedReceipt

call

D3MCoreDeployScript.rawToConvertedReceipts(rawReceipts) [StdCheatsSafe], 341
 D3MCoreDeployScript.readReceipt(path, index) [StdCheatsSafe], 340
 D3MCoreInitScript.rawToConvertedReceipts(rawReceipts) [StdCheatsSafe], 369
 D3MCoreInitScript.readReceipt(path, index) [StdCheatsSafe], 368
 D3MDeployScript.rawToConvertedReceipts(rawReceipts) [StdCheatsSafe], 395
 D3MDeployScript.readReceipt(path, index) [StdCheatsSafe], 394
 D3MInitScript.rawToConvertedReceipts(rawReceipts) [StdCheatsSafe], 415
 D3MInitScript.readReceipt(path, index) [StdCheatsSafe], 414

definition

D3MCoreDeployScript.rawToConvertedReceipt(rawReceipt) [StdCheatsSafe], 341
 D3MCoreInitScript.rawToConvertedReceipt(rawReceipt) [StdCheatsSafe], 369
 D3MDeployScript.rawToConvertedReceipt(rawReceipt) [StdCheatsSafe], 395
 D3MInitScript.rawToConvertedReceipt(rawReceipt) [StdCheatsSafe], 415

rawToConvertedReceiptLogs

call

D3MCoreDeployScript.rawToConvertedReceipt(rawReceipt) [StdCheatsSafe], 341
 D3MCoreInitScript.rawToConvertedReceipt(rawReceipt) [StdCheatsSafe], 369
 D3MDeployScript.rawToConvertedReceipt(rawReceipt) [StdCheatsSafe], 395
 D3MInitScript.rawToConvertedReceipt(rawReceipt) [StdCheatsSafe], 415

definition

D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341
 D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369
 D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395
 D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415

rawToConvertedReceipts

call

D3MCoreDeployScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 339
 D3MCoreDeployScript.readReceipts(path) [StdCheatsSafe], 340
 D3MCoreInitScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 367
 D3MCoreInitScript.readReceipts(path) [StdCheatsSafe], 368
 D3MDeployScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 393
 D3MDeployScript.readReceipts(path) [StdCheatsSafe], 394
 D3MInitScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 413
 D3MInitScript.readReceipts(path) [StdCheatsSafe], 414

definition

D3MCoreDeployScript.rawToConvertedReceipts(rawReceipts) [StdCheatsSafe], 341
 D3MCoreInitScript.rawToConvertedReceipts(rawReceipts) [StdCheatsSafe], 369

D3MDeployScript.rawToConvertedReceipts(rawReceipts) [StdCheatsSafe], 395
 D3MInitScript.rawToConvertedReceipts(rawReceipts) [StdCheatsSafe], 415
 RawTx1559
 definition
 struct StdCheatsSafe.RawTx1559, 331, 359, 385, 405
 RawTx1559Detail
 definition
 struct StdCheatsSafe.RawTx1559Detail, 331, 359, 385, 405
 RAY
 definition
 CalleeMakerOtcDai.RAY [CalleeMakerOtc] : [uint256](#), 549
 ClipperMom.RAY: [uint256](#), 257
 Clipper.RAY: [uint256](#), 241
 CurveLpTokenUniv3Callee.RAY: [uint256](#), 551
 D3MAavePlan.RAY: [uint256](#), 311
 D3MAavePool.RAY: [uint256](#), 349
 D3MHub.RAY: [uint256](#), 319
 D3MInitScript.RAY: [uint256](#), 403
 DssExecLib.RAY: [uint256](#), 729
 DssProxyActionsDsr.RAY [Common] : [uint256](#), 547
 DssProxyActionsEnd.RAY [Common] : [uint256](#), 543
 DssProxyActions.RAY [Common] : [uint256](#), 529
 DssPsm.RAY: [uint256](#), 509
 DssVestSuckable.RAY: [uint256](#), 685
 End.RAY: [uint256](#), 295
 ExponentialDecrease.RAY: [uint256](#), 239
 LinearDecrease.RAY: [uint256](#), 235
 MCD.RAY: [uint256](#), 719
 PSMCallee.RAY: [uint256](#), 555
 StairstepExponentialDecrease.RAY: [uint256](#), 237
 TeleportJoin.RAY: [uint256](#), 641
 UniswapV2CalleeDai.RAY [UniswapV2Callee] : [uint256](#), 557
 UniswapV2LpTokenCalleeDai.RAY [UniswapV2Callee(2)] : [uint256](#), 559
 UniswapV3Callee.RAY: [uint256](#), 567
 WstETHCurveUniv3Callee.RAY: [uint256](#), 563
 read
 CalleeMakerOtcDai.clipperCall(sender, daiAmt, gemAmt, data) [X](#), 550
 Clipper.constructor(vat_, spotter_, dog_, ilk_) [X](#), 243
 ClipperMom.rmul(x, y), 257
 ClipperMom.setPriceTolerance(clip, value) [X](#), 258
 Clipper.rdiv(x, y), 244
 Clipper.rmul(x, y), 244
 CurveLpTokenUniv3Callee.clipperCall(sender, owe, slice, data) [X](#), 552
 D3MAavePlan.rdiv(x, y), 312
 D3MAavePlan.rmul(x, y), 312
 D3MAavePool.rdiv(x, y), 350
 D3MHub.exec(ilk) [X](#), 324
 D3MHub.exec(ilk, _pool, Art, lineWad), 322
 D3MHub.uncull(ilk) [X](#), 326
 D3MHub.wipe(ilk, _pool, urn), 322
 D3MInitScript.run() [X](#), 420
 DssExecLib.rdiv(x, y), 730
 DssExecLib.setDSR(_rate, _doDrip) [X](#), 737
 DssExecLib.setGlobalStabilityFee(_rate) [X](#), 741
 DssExecLib.setIlkStabilityFee(_ilk, _rate, _doDrip) [X](#), 746
 DssProxyActionsDsr.exit(daiJoin, pot, wad) [X](#), 548
 DssProxyActionsDsr.exitAll(daiJoin, pot) [X](#), 548
 DssProxyActionsDsr.join(daiJoin, pot, wad) [X](#), 547
 DssProxyActionsEnd.cashETH(ethJoin, end, ilk, wad) [X](#), 544
 DssProxyActionsEnd.cashGem(gemJoin, end, ilk, wad) [X](#), 545
 DssProxyActions._getDrawDart(vat, jug, urn, ilk, wad), 530
 DssProxyActions._getWipeAllWad(vat, usr, urn, ilk), 530
 DssProxyActions.wipe(manager, daiJoin, cdp, wad) [X](#), 537

DssPsm.buyGem(usr, gemAmt) [X](#), 511
 DssPsm.sellGem(usr, gemAmt) [X](#), 510
 DssVestSuckable.pay(_guy, _amt), 693
 End.flow(ilk) [X](#), 300
 End.pack(wad) [X](#), 301
 End.rmul(x, y), 298
 ExponentialDecrease.file(what, [data](#)) [X](#) [a](#), 239
 ExponentialDecrease.price(top, dur), 240
 ExponentialDecrease.rmul(x, y), 240
 LinearDecrease.price(top, dur), 236
 LinearDecrease.rmul(x, y), 236
 MCD.initIlk(dss, ilk, join, pip), 720
 PSMCallee.clipperCall([sender](#), owe, slice, [data](#)) [X](#), 555
 StairstepExponentialDecrease.file(what, [data](#)) [X](#) [a](#), 237
 StairstepExponentialDecrease.price(top, dur), 238
 StairstepExponentialDecrease.rmul(x, y), 238
 TeleportJoin.cure(), 643
 TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643
 UniswapV2CalleeDai.clipperCall([sender](#), daiAmt, gemAmt, [data](#)) [X](#), 558
 UniswapV2LpTokenCalleeDai.clipperCall([sender](#), daiAmt, gemAmt, [data](#)) [X](#), 560
 UniswapV3Callee.clipperCall([sender](#), owe, slice, [data](#)) [X](#), 567
 WstETHCurveUniv3Callee.clipperCall([sender](#), owe, slice, [data](#)) [X](#), 564

rdiv

call

Clipper.getFeedPrice(), 244
 Clipper.status(tic, top), 248
 D3MAavePlan._calculateTargetSupply(targetInterestRate, totalDebt), 313
 D3MAavePool.deposit(wad) [X](#), 350
 DssExecLib.setAuctionPermittedDrop(_ilk, _pct_bps) [X](#), 745
 DssExecLib.setD3MTargetInterestRate(_d3m, _pct_bps) [X](#), 749
 DssExecLib.setDAIReferenceValue(_value) [X](#), 741
 DssExecLib.setExponentialDecrease(_calc, _pct_bps) [X](#), 747
 DssExecLib.setIlkLiquidationRatio(_ilk, _pct_bps) [X](#), 744
 DssExecLib.setLiquidationBreakerPriceTolerance(_clip, _pct_bps) [X](#), 745
 DssExecLib.setStairstepExponentialDecrease(_calc, _duration, _pct_bps) [X](#), 746
 DssExecLib.setStartingPriceMultiplicativeFactor(_ilk, _pct_bps) [X](#), 744
 Spotter.poke(ilk) [X](#), 228

definition

Clipper.rdiv(x, y), 244
 D3MAavePlan.rdiv(x, y), 312
 D3MAavePool.rdiv(x, y), 350
 DssExecLib.rdiv(x, y), 730
 Spotter.rdiv(x, y), 228

read

call

End.cage(ilk) [X](#), 299

definition

D3MOracle.read(), 354
 Median.read(), 282
 OSM.read(), 288

readAddress

call

D3MCoreDeployScript.run() [X](#), 346
 D3MCoreInitScript.run() [X](#), 374
 D3MDeployScript.run() [X](#), 400
 D3MInitScript.run() [X](#), 420
 ScriptTools.readAddress(json, key, envKey), 717

definition

ScriptTools.readAddress(json, key, envKey), 717

readAddressArray

call

ScriptTools.readAddressArray(json, key, envKey), 718

definition


```
ScriptTools.readAddressArray(json, key, envKey), 718
readBool
  call
    D3MInitScript.run() X, 420
    ScriptTools.readBool(json, key, envKey), 718
  definition
    ScriptTools.readBool(json, key, envKey), 718
readBoolArray
  call
    ScriptTools.readBoolArray(json, key, envKey), 718
  definition
    ScriptTools.readBoolArray(json, key, envKey), 718
readBytes
  call
    ScriptTools.readBytes(json, key, envKey), 718
  definition
    ScriptTools.readBytes(json, key, envKey), 718
readBytes32
  call
    ScriptTools.readBytes32(json, key, envKey), 717
  definition
    ScriptTools.readBytes32(json, key, envKey), 717
readBytes32Array
  call
    ScriptTools.readBytes32Array(json, key, envKey), 717
  definition
    ScriptTools.readBytes32Array(json, key, envKey), 717
readBytesArray
  call
    ScriptTools.readBytesArray(json, key, envKey), 718
  definition
    ScriptTools.readBytesArray(json, key, envKey), 718
readEIP1559ScriptArtifact
  definition
    D3MCoreDeployScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 339
    D3MCoreInitScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 367
    D3MDeployScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 393
    D3MInitScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 413
readFile
  call
    D3MCoreDeployScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 339
    D3MCoreDeployScript.readReceipt(path, index) [StdCheatsSafe], 340
    D3MCoreDeployScript.readReceipts(path) [StdCheatsSafe], 340
    D3MCoreDeployScript.readTx1559(path, index) [StdCheatsSafe], 340
    D3MCoreDeployScript.readTx1559s(path) [StdCheatsSafe], 340
    D3MCoreInitScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 367
    D3MCoreInitScript.readReceipt(path, index) [StdCheatsSafe], 368
    D3MCoreInitScript.readReceipts(path) [StdCheatsSafe], 368
    D3MCoreInitScript.readTx1559(path, index) [StdCheatsSafe], 368
    D3MCoreInitScript.readTx1559s(path) [StdCheatsSafe], 368
    D3MDeployScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 393
    D3MDeployScript.readReceipt(path, index) [StdCheatsSafe], 394
    D3MDeployScript.readReceipts(path) [StdCheatsSafe], 394
    D3MDeployScript.readTx1559(path, index) [StdCheatsSafe], 394
    D3MDeployScript.readTx1559s(path) [StdCheatsSafe], 394
    D3MInitScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 413
    D3MInitScript.readReceipt(path, index) [StdCheatsSafe], 414
    D3MInitScript.readReceipts(path) [StdCheatsSafe], 414
    D3MInitScript.readTx1559(path, index) [StdCheatsSafe], 414
    D3MInitScript.readTx1559s(path) [StdCheatsSafe], 414
    ScriptTools.readInput(name), 713
    ScriptTools.readOutput(name), 714
    ScriptTools.readOutput(name, timestamp), 713
```

readInput

call

ScriptTools.loadConfig(), 714
 ScriptTools.loadConfig(name), 714

definition

ScriptTools.readInput(name), 713

readInt

call

ScriptTools.readInt(json, key, envKey), 717

definition

ScriptTools.readInt(json, key, envKey), 717

readIntArray

call

ScriptTools.readIntArray(json, key, envKey), 717

definition

ScriptTools.readIntArray(json, key, envKey), 717

readOutput

call

ScriptTools.loadDependencies(), 715
 ScriptTools.loadDependencies(name), 714

definition

ScriptTools.readOutput(name), 714
 ScriptTools.readOutput(name, **timestamp**), 713

readReceipt

definition

D3MCoreDeployScript.readReceipt(path, index) [StdCheatsSafe], 340
 D3MCoreInitScript.readReceipt(path, index) [StdCheatsSafe], 368
 D3MDeployScript.readReceipt(path, index) [StdCheatsSafe], 394
 D3MInitScript.readReceipt(path, index) [StdCheatsSafe], 414

readReceipts

definition

D3MCoreDeployScript.readReceipts(path) [StdCheatsSafe], 340
 D3MCoreInitScript.readReceipts(path) [StdCheatsSafe], 368
 D3MDeployScript.readReceipts(path) [StdCheatsSafe], 394
 D3MInitScript.readReceipts(path) [StdCheatsSafe], 414

readString

call

D3MDeployScript.run() **X**, 400
 D3MInitScript.run() **X**, 420
 ScriptTools.readString(json, key, envKey), 717

definition

ScriptTools.readString(json, key, envKey), 717

readStringArray

call

ScriptTools.readStringArray(json, key, envKey), 717

definition

ScriptTools.readStringArray(json, key, envKey), 717

readTx1559

definition

D3MCoreDeployScript.readTx1559(path, index) [StdCheatsSafe], 340
 D3MCoreInitScript.readTx1559(path, index) [StdCheatsSafe], 368
 D3MDeployScript.readTx1559(path, index) [StdCheatsSafe], 394
 D3MInitScript.readTx1559(path, index) [StdCheatsSafe], 414

readTx1559s

definition

D3MCoreDeployScript.readTx1559s(path) [StdCheatsSafe], 340
 D3MCoreInitScript.readTx1559s(path) [StdCheatsSafe], 368
 D3MDeployScript.readTx1559s(path) [StdCheatsSafe], 394
 D3MInitScript.readTx1559s(path) [StdCheatsSafe], 414

readUInt

call

D3MInitScript.run() **X**, 420
 ScriptTools.readUInt(json, key, envKey), 716

[definition](#)
 ScriptTools.readUint(json, key, envKey), 716
 readUintArray
 [call](#)
 ScriptTools.readUintArray(json, key, envKey), 717
 [definition](#)
 ScriptTools.readUintArray(json, key, envKey), 717
 Receipt
 [definition](#)
 struct StdCheatsSafe.Receipt, 333, 361, 387, 407
 ReceiptLog
 [definition](#)
 struct StdCheatsSafe.ReceiptLog, 334, 362, 388, 408
 reclaim
 [definition](#)
 L1GovernanceRelay.reclaim(receiver, amount) **X** **a**, 579
 recover
 [call](#)
 Median.poke(val_, age_, v, r, s) **X**, 283
 [definition](#)
 Median.recover(val_, age_, v, r, s), 283
 redeemable
 [definition](#)
 D3MAavePool.redeemable(), 352
 D3MCompoundPool.redeemable(), 380
 redeemUnderlying
 [call](#)
 D3MCompoundPool.withdraw(wad) **X**, 379
 redo
 [definition](#)
 Clipper.redo(id, kpr) **X**, 245
 refuse
 [definition](#)
 L1EscrowMom.refuse(spender) **X** **a**, 638
 reg
 [call](#)
 DssExecLib.addCollateralBase(_ilk, _gem, _join, _clip, _calc, _pip) **X**, 749
 DssExecLib.clip(_ilk), 732
 DssExecLib.flip(_ilk), 732
 [definition](#)
 DssExecLib.reg(), 731
 relay
 [definition](#)
 BasicRelay.relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s) **X**, 658
 L1GovernanceRelay.relay(spell) **X** **a**, 621
 L1GovernanceRelay.relay(target, targetData, l1CallValue, maxGas, gasPriceBid, maxSubmissionCost) **X** **a**, 579
 L1GovernanceRelay.relay(target, targetData, l2gas) **X** **a**, 601
 L2GovernanceRelay.relay(target, targetData) **X**, 586, 611
 TrustedRelay.relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s) **X**, 663
 relayers
 write
 BasicRelay.addRelayers(relayers_) **X** **a**, 657
 BasicRelay.removeRelayers(relayers_) **X** **a**, 658
 TrustedRelay.addRelayers(relayers_) **X** **a**, 662
 TrustedRelay.removeRelayers(relayers_) **X** **a**, 662
 [definition](#)
 BasicRelay.relayers: **mapping**(address => uint256), 657
 TrustedRelay.relayers: **mapping**(address => uint256), 661
 read

```

BasicRelay.relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v,
r, s) X, 658
TrustedRelay.requestMint(teleportGUID, signatures, maxFeePercentage, gasFee,
expiry, v, r, s), 664
RELAY_SELECTOR
definition
L1GovernanceRelay.RELAY_SELECTOR: uint256, 621
read
L1GovernanceRelay.relay(spell) X a, 621
rely
call
D3MInit.initAave(dss, d3m, cfg, aaveCfg), 726
D3MInit.initCompound(dss, d3m, cfg, compoundCfg), 727
D3MInit.initCore(dss, d3mCore), 725
DssExecLib.authorize(_base, _ward) X, 733
DssExecLib.linearInterpolation(_name, _target, _ilk, _what, _startTime, _start,
_end, _duration) X, 752
DssExecLib.linearInterpolation(_name, _target, _what, _startTime, _start, _end,
_duration) X, 751
MCD.initIlk(dss, ilk, join, pip), 720
ScriptTools.switchOwner(base, deployer, newOwner), 716
definition
AuthGemJoin5.rely(usr) X a, 503
AuthGemJoin8.rely(usr) X a, 505
AuthGemJoin.rely(usr) X a, 447, 507
BasicRelay.rely(usr) X a, 657
ChainLog.rely(usr) X a, 709
Clipper.rely(usr) X a, 242
Cure.rely(usr) X a, 292
D3MAavePlan.rely(usr) X a, 312
D3MAavePool.rely(usr) X a, 350
D3MCompoundPlan.rely(usr) X a, 316
D3MCompoundPool.rely(usr) X a, 378
D3MHub.rely(usr) X a, 321
D3MOracle.rely(usr) X a, 353
DaiJoin.rely(usr) X a, 425
Dai.rely(guy) X a, 223
Dai.rely(usr) X a, 587, 613
Dog.rely(usr) X a, 252
DssAutoLine.rely(usr) X a, 707
DssPsm.rely(usr) X a, 509
DssVestMintable.rely(_usr) [DssVest] X a, 677
DssVestSuckable.rely(_usr) [DssVest] X a, 687
DssVestTransferrable.rely(_usr) [DssVest] X a, 697
End.rely(usr) X a, 297
ESM.rely(usr) X a, 304
ExponentialDecrease.rely(usr) X a, 239
Flapper.rely(usr) X a, 262
Flopper.rely(usr) X a, 265
GemJoin2.rely(usr) X a, 427
GemJoin3.rely(usr) X a, 429
GemJoin4.rely(usr) X a, 433
GemJoin5.rely(usr) X a, 435
GemJoin6.rely(usr) X a, 437
GemJoin7.rely(usr) X a, 439
GemJoin8.rely(usr) X a, 441
GemJoin9.rely(usr) X a, 443
GemJoin.rely(usr) X a, 424
IlkRegistry.rely(usr) X a, 667
Jug.rely(usr) X a, 273
L1DAIBridge.rely(usr) X a, 618
L1DaiGateway.rely(usr) X a, 570
L1DAITokenBridge.rely(usr) X a, 593

```

- L1Escrow.rely(usr) [X a](#), 575, 597, 635
- L1GovernanceRelay.rely(usr) [X a](#), 578, 601, 621
- L2DaiGateway.rely(usr) [X a](#), 582
- L2DAITokenBridge.rely(usr) [X a](#), 605
- LinearDecrease.rely(usr) [X a](#), 235
- ManagedGemJoin.rely(usr) [X a](#), 449
- Median.rely(usr) [X a](#), 282
- OSM.rely(usr) [X a](#), 286
- Pot.rely(guy) [X a](#), 277
- Spotter.rely(guy) [X a](#), 227
- StairstepExponentialDecrease.rely(usr) [X a](#), 237
- TeleportJoin.rely(usr) [X a](#), 642
- TeleportOracleAuth.rely(usr) [X a](#), 649
- TeleportRouter.rely(usr) [X a](#), 653
- TrustedRelay.rely(usr) [X a](#), 662
- Vat.rely(usr) [X a](#), 229
- Vow.rely(usr) [X a](#), 269

rememberKey

- [call](#)
 - D3MCoreDeployScript.deriveRememberKey(mnemonic, index) [StdCheatsSafe], 343
 - D3MCoreInitScript.deriveRememberKey(mnemonic, index) [StdCheatsSafe], 371
 - D3MDeployScript.deriveRememberKey(mnemonic, index) [StdCheatsSafe], 397
 - D3MInitScript.deriveRememberKey(mnemonic, index) [StdCheatsSafe], 417

remIlk

- [call](#)
 - DssExecLib.removeIlkFromAutoLine(_ilk) [X](#), 743
- [definition](#)
 - DssAutoLine.remIlk(ilk) [X a](#), 706

_remove

- [call](#)
 - Clipper.take(id, amt, max, who, [data](#)) [X](#), 246
 - Clipper.yank(id) [X a](#), 249
 - EnumerableSet.remove(set, [value](#)), 758–760
 - IlkRegistry.remove(ilk) [X](#), 669
 - IlkRegistry.removeAuth(ilk) [X a](#), 669
 - TeleportRouter.file(what, domain, [data](#)) [X a](#), 653
- [definition](#)
 - Clipper._remove(id), 247
 - EnumerableSet._remove(set, [value](#)), 756, 758–760
 - IlkRegistry._remove(ilk), 670
 - IlkRegistry.remove(ilk) [X](#), 669

_removeAddress

- [call](#)
 - ChainLog.removeAddress(_key) [X a](#), 710
- [definition](#)
 - ChainLog._removeAddress(_key), 711
 - ChainLog.removeAddress(_key) [X a](#), 710

removeAuth

- [definition](#)
 - IlkRegistry.removeAuth(ilk) [X a](#), 669

removed

- [write](#)
 - D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341
 - D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369
 - D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395
 - D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415

read

- D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341
- D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369
- D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395
- D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415

removeIlkFromAutoLine

- [definition](#)

DssExecLib.removeIlkFromAutoLine(_ilk) **X**, 743
 removeLiquidity
 call
 UniswapV2LpTokenCalleeDai.clipperCall(sender, daiAmt, gemAmt, data) **X**, 560
 removeLiquidityOneCoin
 call
 CurveLpTokenUniv3Callee.clipperCall(sender, owe, slice, data) **X**, 552
 removeReaderFromWhitelist
 definition
 DssExecLib.removeReaderFromWhitelist(_oracle, _reader) **X**, 747
 removeReaderFromWhitelistCall
 definition
 DssExecLib.removeReaderFromWhitelistCall(_oracle, _reader) **X**, 748
 removeRelayers
 definition
 BasicRelay.removeRelayers(relayers_) **X** a, 658
 TrustedRelay.removeRelayers(relayers_) **X** a, 662
 removeSigners
 definition
 TeleportOracleAuth.removeSigners(signers_) **X** a, 650
 TrustedRelay.removeSigners(signers_) **X**, 663
 RENBTC
 definition
 contract RENBTC, 483
 REP
 definition
 contract REP, 485
 requestMint
 call
 BasicRelay.relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s) **X**, 658
 LiDAITeleportGateway.finalizeRegisterTeleport(teleport) **X**, 623
 TeleportOracleAuth.requestMint(teleportGUID, signatures, maxFeePercentage, operatorFee) **X**, 650
 TeleportRouter.requestMint(teleportGUID, maxFeePercentage, operatorFee) **X**, 654
 TrustedRelay.relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s) **X**, 663
 TrustedRelay.requestMint(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s), 664
 definition
 TeleportJoin.requestMint(teleportGUID, maxFeePercentage, operatorFee) **X** a, 644
 TeleportOracleAuth.requestMint(teleportGUID, signatures, maxFeePercentage, operatorFee) **X**, 650
 TeleportRouterMock.requestMint(teleportGUID, maxFeePercentage, operatorFee) **X**, 633
 TeleportRouter.requestMint(teleportGUID, maxFeePercentage, operatorFee) **X**, 654
 TrustedRelay.requestMint(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s), 664
 res
 write
 DssVestMintable.restrict(_id) [DssVest] **X**, 681
 DssVestMintable.unrestrict(_id) [DssVest] **X**, 681
 DssVestSuckable.restrict(_id) [DssVest] **X**, 690
 DssVestSuckable.unrestrict(_id) [DssVest] **X**, 691
 DssVestTransferrable.restrict(_id) [DssVest] **X**, 701
 DssVestTransferrable.unrestrict(_id) [DssVest] **X**, 701
 definition
 DssVestMintable.res(_id) [DssVest], 677
 DssVestSuckable.res(_id) [DssVest], 686
 DssVestTransferrable.res(_id) [DssVest], 696
 read
 DssVestMintable.res(_id) [DssVest], 677
 DssVestSuckable.res(_id) [DssVest], 686
 DssVestTransferrable.res(_id) [DssVest], 696

restrict

definition

DssVestMintable.restrict(_id) [DssVest] [X](#), 681
 DssVestSuckable.restrict(_id) [DssVest] [X](#), 690
 DssVestTransferrable.restrict(_id) [DssVest] [X](#), 701

resumeGasMetering

call

D3MCoreDeployScript.modifier noGasMetering() [StdCheatsSafe], 335
 D3MCoreInitScript.modifier noGasMetering() [StdCheatsSafe], 363
 D3MDeployScript.modifier noGasMetering() [StdCheatsSafe], 389
 D3MInitScript.modifier noGasMetering() [StdCheatsSafe], 409

revokesGovernanceAccess

definition

ESM.revokesGovernanceAccess(), 303

REWARD_TOKEN

call

D3MAavePool.collect() [X](#), 352

rho

write

Jug.drip(ilk) [X](#), 275
 Pot.constructor(vat_) [X](#), 277
 Pot.drip() [X](#), 279

definition

Pot.rho: [uint256](#), 277

read

Jug.drip(ilk) [X](#), 275
 Jug.file(ilk, what, [data](#)) [X](#) [a](#), 275
 Pot.drip() [X](#), 279
 Pot.file(what, [data](#)) [X](#) [a](#), 278
 Pot.join(wad) [X](#), 279

_rmul

call

Clipper.kick(tab, lot, usr, kpr) [X](#) [a](#), 244
 ClipperMom.tripBreaker(clip) [X](#), 259
 Clipper.redo(id, kpr) [X](#), 245
 D3MAavePlan._calculateTargetSupply(targetInterestRate, totalDebt), 313
 End.cash(ilk, wad) [X](#), 301
 End.flow(ilk) [X](#), 300
 End.skim(ilk, urn) [X](#), 300
 ExponentialDecrease.price(top, dur), 240
 Jug.drip(ilk) [X](#), 275
 LinearDecrease.price(top, dur), 236
 Pot.drip() [X](#), 279
 StairstepExponentialDecrease.price(top, dur), 238

definition

ClipperMom._rmul(x, y), 257
 Clipper._rmul(x, y), 244
 D3MAavePlan._rmul(x, y), 312
 End._rmul(x, y), 298
 ExponentialDecrease._rmul(x, y), 240
 Jug._rmul(x, y), 274
 LinearDecrease._rmul(x, y), 236
 Pot._rmul(x, y), 278
 StairstepExponentialDecrease._rmul(x, y), 238

rpcUrl

call

D3MCoreDeployScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 345
 D3MCoreInitScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 373
 D3MDeployScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 399
 D3MInitScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 419

_rpow

call

ExponentialDecrease.price(top, dur), 240

Jug.drip(ilk) [X](#), 275
 Pot.drip() [X](#), 279
 StairstepExponentialDecrease.price(top, dur), 238

definition

ExponentialDecrease.rpow(x, n, b), 240
 Jug.rpow(x, n, b), 274
 Pot.rpow(x, n, base), 277
 StairstepExponentialDecrease.rpow(x, n, b), 238

run

definition

D3MCoreDeployScript.run() [X](#), 346
 D3MCoreInitScript.run() [X](#), 374
 D3MDeployScript.run() [X](#), 400
 D3MInitScript.run() [X](#), 420

rxid

write

DssVestMintable._vest(_id, _maxAmt) [DssVest], 679
 DssVestSuckable._vest(_id, _maxAmt) [DssVest], 689
 DssVestTransferrable._vest(_id, _maxAmt) [DssVest], 699

definition

DssVestMintable.rxd(_id) [DssVest], 677
 DssVestSuckable.rxd(_id) [DssVest], 687
 DssVestTransferrable.rxd(_id) [DssVest], 697

read

DssVestMintable.rxd(_id) [DssVest], 677
 DssVestMintable.valid(_id) [DssVest], 682
 DssVestSuckable.rxd(_id) [DssVest], 687
 DssVestSuckable.valid(_id) [DssVest], 692
 DssVestTransferrable.rxd(_id) [DssVest], 697
 DssVestTransferrable.valid(_id) [DssVest], 702

S

safeLockETH

definition

DssProxyActions.safeLockETH(manager, ethJoin, cdp, owner) **X**, 534

safeLockGem

definition

DssProxyActions.safeLockGem(manager, gemJoin, cdp, amt, transferFrom, owner) **X**, 535

SafeMath

definition

library SafeMath, 753

SAFEMAX

definition

D3MHub.SAFEMAX: **uint256**, 319

read

D3MHub._exec(ilk, _pool, Art, lineWad), 322

safeWipe

definition

DssProxyActions.safeWipe(manager, daiJoin, cdp, wad, owner) **X**, 537

safeWipeAll

definition

DssProxyActions.safeWipeAll(manager, daiJoin, cdp, owner) **X**, 538

Sale

definition

struct Clipper.Sale, 242

sales

call

End.snip(ilk, id) **X**, 299

write

Clipper.kick(tab, lot, usr, kpr) **X** **a**, 244Clipper.redo(id, kpr) **X**, 245

Clipper.remove(id), 247

Clipper.take(id, amt, max, who, **data**) **X**, 246

definition

Clipper.sales: **mapping**(**uint256** => Clipper.Sale), 241

read

Clipper.getStatus(id), 248

Clipper.kick(tab, lot, usr, kpr) **X** **a**, 244Clipper.redo(id, kpr) **X**, 245

Clipper.remove(id), 247

Clipper.take(id, amt, max, who, **data**) **X**, 246Clipper.yank(id) **X** **a**, 249

say

write

Cure.load(src) **X**, 293

definition

Cure.say: **uint256**, 291

read

Cure.load(src) **X**, 293

Cure.tell(), 292

scaledBalanceOf

call

D3MAavePool.deposit(wad) **X**, 350

schedule

definition

DssExec.schedule() **X**, 309

ScriptTools

definition

library ScriptTools, 713

sellAllAmount

call

CalleeMakerOtcDai.clipperCall(**sender**, daiAmt, gemAmt, **data**) **X**, 550

sellGem

call
 PSMCallee.clipperCall(sender, owe, slice, data) X, 555

definition
 DssPsm.sellGem(usr, gemAmt) X, 510

sendCrossDomainMessage
 call
 L1DAITokenBridge.initiateERC20Deposit(_from, _to, _amount, _l2Gas, _data), 594
 L1GovernanceRelay.relay(target, targetData, l2gas) X a, 601
 L2DAITokenBridge.initiateWithdrawal(_from, _to, _amount, _l1Gas, _data), 606

definition
 L1DAITokenBridge.sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message) [OVM_CrossDomainEnabled], 593
 L1GovernanceRelay.sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message) [OVM_CrossDomainEnabled], 600
 L2DAITokenBridge.sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message) [OVM_CrossDomainEnabled], 604
 L2GovernanceRelay.sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message) [OVM_CrossDomainEnabled], 610

sendMessage
 call
 L1DAITokenBridge.sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message) [OVM_CrossDomainEnabled], 593
 L1GovernanceRelay.sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message) [OVM_CrossDomainEnabled], 600
 L2DAITokenBridge.sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message) [OVM_CrossDomainEnabled], 604
 L2GovernanceRelay.sendCrossDomainMessage(_crossDomainTarget, _gasLimit, _message) [OVM_CrossDomainEnabled], 610

sendMessageToL2
 call
 L1DAIBridge.deposit(amount, l2Recipient) X, 619
 L1DAIBridge.forceWithdrawal(amount, l2Sender) X, 620
 L1GovernanceRelay.relay(spell) X a, 621

sendPaymentFromSurplusBuffer
 definition
 DssExecLib.sendPaymentFromSurplusBuffer(_target, _amount) X, 751

sendTxToL1
 call
 L2DaiGateway.outboundTransfer(l1Token, to, amount, , , data) X, 583
 L2DaiGateway.sendTxToL1(user, to, data) [L2CrossDomainEnabled], 581
 L2GovernanceRelay.sendTxToL1(user, to, data) [L2CrossDomainEnabled], 585

definition
 L2DaiGateway.sendTxToL1(user, to, data) [L2CrossDomainEnabled], 581
 L2GovernanceRelay.sendTxToL1(user, to, data) [L2CrossDomainEnabled], 585

sendTxToL2
 call
 L1DaiGateway.outboundTransfer(l1Token, to, amount, maxGas, gasPriceBid, data) X, 571

definition
 L1DaiGateway.sendTxToL2(target, user, maxSubmissionCost, maxGas, gasPriceBid, data) [L1CrossDomainEnabled], 570
 L1GovernanceRelay.sendTxToL2(target, user, maxSubmissionCost, maxGas, gasPriceBid, data) [L1CrossDomainEnabled], 577

sendTxToL2NoAliasing
 call
 L1GovernanceRelay.relay(target, targetData, l1CallValue, maxGas, gasPriceBid, maxSubmissionCost) X a, 579

definition
 L1DaiGateway.sendTxToL2NoAliasing(target, user, l1CallValue, maxSubmissionCost, maxGas, gasPriceBid, data) [L1CrossDomainEnabled], 570
 L1GovernanceRelay.sendTxToL2NoAliasing(target, user, l1CallValue, maxSubmissionCost, maxGas, gasPriceBid, data) [L1CrossDomainEnabled], 578

sendValue

```

    definition
        Address.sendValue(recipient, amount), 763
serializeAddress
    call
        ScriptTools.exportContract(name, label, addr), 715
Set
    definition
        struct EnumerableSet.Set, 755
setAddress
    call
        ChainLog.constructor() X, 709
        D3MInit.init(dss, d3m, cfg, gem), 725
        D3MInit.initCore(dss, d3mCore), 725
        DssExecLib.setChangelogAddress(_key, _val) X, 733
    definition
        ChainLog.setAddress(_key, _addr) X a, 710
setAuctionPermittedDrop
    call
        DssExecLib.addNewCollateral(co) X, 750
    definition
        DssExecLib.setAuctionPermittedDrop(_ilk, _pct.bps) X, 745
setAuctionTimeBeforeReset
    call
        DssExecLib.addNewCollateral(co) X, 750
    definition
        DssExecLib.setAuctionTimeBeforeReset(_ilk, _duration) X, 745
setAuthority
    call
        D3MInit.initCore(dss, d3mCore), 725
        DssExecLib.setAuthority(_base, _authority) X, 734
    definition
        ClipperMom.setAuthority(authority_) X, 258
        D3MMom.setAuthority(authority_) X, 356
        DssExecLib.setAuthority(_base, _authority) X, 734
        L1EscrowMom.setAuthority(authority_) X, 638
setBalance
    call
        GodMode.setBalance(token, who, amount), 769
    definition
        GodMode.setBalance(token, who, amount), 768, 769
setBar
    call
        DssExecLib.setMedianWritersQuorum(_median, _minQuorum) X, 748
    definition
        Median.setBar(bar_) X a, 284
setBreaker
    definition
        ClipperMom.setBreaker(clip, level, delay) X a, 258
setCeiling
    definition
        L1DAIBridge.setCeiling(_ceiling) X a, 619
setChain
    call
        D3MCoreDeployScript.setChain(chainAlias, chain) [StdChains], 344
        D3MCoreDeployScript.setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains], 346
        D3MCoreInitScript.setChain(chainAlias, chain) [StdChains], 372
        D3MCoreInitScript.setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains], 374
        D3MDeployScript.setChain(chainAlias, chain) [StdChains], 398
        D3MDeployScript.setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains], 400
        D3MInitScript.setChain(chainAlias, chain) [StdChains], 419
        D3MInitScript.setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains], 420
    definition
        D3MCoreDeployScript.setChain(chainAlias, chain) [StdChains], 344

```

D3MCoreInitScript.setChain(chainAlias, chain) [StdChains], 372
 D3MDeployScript.setChain(chainAlias, chain) [StdChains], 398
 D3MInitScript.setChain(chainAlias, chain) [StdChains], 418, 419
 setChainWithDefaultRpcUrl
 call
 D3MCoreDeployScript.initialize() [StdChains], 345
 D3MCoreInitScript.initialize() [StdChains], 373
 D3MDeployScript.initialize() [StdChains], 399
 D3MInitScript.initialize() [StdChains], 419
 definition
 D3MCoreDeployScript.setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains], 346
 D3MCoreInitScript.setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains], 374
 D3MDeployScript.setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains], 400
 D3MInitScript.setChainWithDefaultRpcUrl(chainAlias, chain) [StdChains], 420
 setChangelogAddress
 definition
 DssExecLib.setChangelogAddress(_key, _val) X, 733
 setChangelogIPFS
 definition
 DssExecLib.setChangelogIPFS(_ipfsHash) X, 733
 setChangelogSHA256
 definition
 DssExecLib.setChangelogSHA256(_SHA256Sum) X, 733
 setChangelogVersion
 definition
 DssExecLib.setChangelogVersion(_version) X, 733
 setContract
 call
 DssExecLib.addCollateralBase(_ilk, _gem, _join, _clip, _calc, _pip) X, 749
 definition
 DssExecLib.setContract(_base, _ilk, _what, _addr) X, 736
 DssExecLib.setContract(_base, _what, _addr) X, 736
 setD3MTargetInterestRate
 definition
 DssExecLib.setD3MTargetInterestRate(_d3m, _pct_bps) X, 749
 setDAIReferenceValue
 definition
 DssExecLib.setDAIReferenceValue(_value) X, 741
 setDebtAuctionBidDuration
 definition
 DssExecLib.setDebtAuctionBidDuration(_duration) X, 739
 setDebtAuctionDAIAmount
 definition
 DssExecLib.setDebtAuctionDAIAmount(_amount) X, 739
 setDebtAuctionDelay
 definition
 DssExecLib.setDebtAuctionDelay(_duration) X, 739
 setDebtAuctionDuration
 definition
 DssExecLib.setDebtAuctionDuration(_duration) X, 740
 setDebtAuctionMKRAmount
 definition
 DssExecLib.setDebtAuctionMKRAmount(_amount) X, 739
 setDebtAuctionMKRIncreaseRate
 definition
 DssExecLib.setDebtAuctionMKRIncreaseRate(_pct_bps) X, 740
 setDSR
 definition
 DssExecLib.setDSR(_rate, _doDrip) X, 737
 setEmergencyShutdownProcessingTime
 definition
 DssExecLib.setEmergencyShutdownProcessingTime(_duration) X, 740
 setExponentialDecrease

```

    definition
        DssExecLib.setExponentialDecrease(_calc, _pct.bps) X, 747
setFeeRate
    call
        PAXG.constructor(supply, fee) X, 478
    definition
        PAXG.setFeeRate(_newFeeRate), 479
setGlobalDebtCeiling
    definition
        DssExecLib.setGlobalDebtCeiling(_amount) X, 737
setGlobalStabilityFee
    definition
        DssExecLib.setGlobalStabilityFee(_rate) X, 741
setIlk
    call
        D3MInit.init(dss, d3m, cfg, gem), 725
        DssExecLib.setIlkAutoLineDebtCeiling(_ilk, _amount) X, 743
        DssExecLib.setIlkAutoLineParameters(_ilk, _amount, _gap, _ttl) X, 742
    definition
        DssAutoLine.setIlk(ilk, line, gap, ttl) X a, 706
setIlkAutoLineDebtCeiling
    definition
        DssExecLib.setIlkAutoLineDebtCeiling(_ilk, _amount) X, 743
setIlkAutoLineParameters
    definition
        DssExecLib.setIlkAutoLineParameters(_ilk, _amount, _gap, _ttl) X, 742
setIlkDebtCeiling
    call
        DssExecLib.addNewCollateral(co) X, 750
        DssExecLib.setRWAILkDebtCeiling(_ilk, _ceiling, _price) X, 742
    definition
        DssExecLib.setIlkDebtCeiling(_ilk, _amount) X, 741
setIlkLiquidationPenalty
    call
        DssExecLib.addNewCollateral(co) X, 750
    definition
        DssExecLib.setIlkLiquidationPenalty(_ilk, _pct.bps) X, 743
setIlkLiquidationRatio
    call
        DssExecLib.addNewCollateral(co) X, 750
    definition
        DssExecLib.setIlkLiquidationRatio(_ilk, _pct.bps) X, 744
setIlkMaxLiquidationAmount
    call
        DssExecLib.addNewCollateral(co) X, 750
    definition
        DssExecLib.setIlkMaxLiquidationAmount(_ilk, _amount) X, 744
setIlkMinVaultAmount
    call
        DssExecLib.addNewCollateral(co) X, 750
    definition
        DssExecLib.setIlkMinVaultAmount(_ilk, _amount) X, 743
setIlkStabilityFee
    call
        DssExecLib.addNewCollateral(co) X, 750
    definition
        DssExecLib.setIlkStabilityFee(_ilk, _rate, _doDrip) X, 746
setImplementation
    call
        AuthGemJoin8.constructor(vat_, ilk_, gem_) X, 505
        GemJoin6.constructor(vat_, ilk_, gem_) X, 437
        GemJoin7.constructor(vat_, ilk_, gem_) X, 439
        GemJoin8.constructor(vat_, ilk_, gem_) X, 441

```

GUSD.[constructor](#)(supply) [X](#), 463
 TUSD.[constructor](#)(supply) [X](#), 487
[definition](#)
 AuthGemJoin8.setImplementation(implementation, permitted) [X](#) [a](#), 506
 GemJoin6.setImplementation(implementation, permitted) [X](#) [a](#), 438
 GemJoin7.setImplementation(implementation, permitted) [X](#) [a](#), 440
 GemJoin8.setImplementation(implementation, permitted) [X](#) [a](#), 442
 GUSD.setImplementation(newImplementation) [X](#), 463
 TUSD.setImplementation(newImplementation) [X](#), 487
 setIPFS
 [call](#)
 DssExecLib.setChangelogIPFS(_ipfsHash) [X](#), 733
 [definition](#)
 ChainLog.setIPFS(_ipfs) [X](#) [a](#), 710
 setKeeperIncentiveFlatRate
 [call](#)
 DssExecLib.addNewCollateral(co) [X](#), 750
 [definition](#)
 DssExecLib.setKeeperIncentiveFlatRate(_ilk, _amount) [X](#), 745
 setKeeperIncentivePercent
 [call](#)
 DssExecLib.addNewCollateral(co) [X](#), 750
 [definition](#)
 DssExecLib.setKeeperIncentivePercent(_ilk, _pct_bps) [X](#), 745
 setLinearDecrease
 [definition](#)
 DssExecLib.setLinearDecrease(_calc, _duration) [X](#), 746
 setLiquidationBreakerPriceTolerance
 [call](#)
 DssExecLib.addNewCollateral(co) [X](#), 750
 [definition](#)
 DssExecLib.setLiquidationBreakerPriceTolerance(_clip, _pct_bps) [X](#), 745
 setMaxDeposit
 [definition](#)
 L1DAIBridge.setMaxDeposit(_maxDeposit) [X](#) [a](#), 619
 setMaxTotalDAILiquidationAmount
 [definition](#)
 DssExecLib.setMaxTotalDAILiquidationAmount(_amount) [X](#), 740
 setMaxTotalDAILiquidationAmountLEGACY
 [definition](#)
 DssExecLib.setMaxTotalDAILiquidationAmountLEGACY(_amount) [X](#), 740
 setMedianWritersQuorum
 [definition](#)
 DssExecLib.setMedianWritersQuorum(_median, _minQuorum) [X](#), 748
 setMinDebtAuctionBidIncrease
 [definition](#)
 DssExecLib.setMinDebtAuctionBidIncrease(_pct_bps) [X](#), 739
 setMinSurplusAuctionBidIncrease
 [definition](#)
 DssExecLib.setMinSurplusAuctionBidIncrease(_pct_bps) [X](#), 738
 setOsm
 [call](#)
 DssExecLib.allowOSMFreeze(_osm, _ilk) [X](#), 748
 setOwner
 [call](#)
 D3MDeploy.deployCore(deployer, owner, daiJoin), 723
 [definition](#)
 ClipperMom.setOwner(owner_) [X](#), 258
 D3MMom.setOwner(owner_) [X](#), 355
 L1EscrowMom.setOwner(owner_) [X](#), 637
 setPriceTolerance
 [call](#)
 DssExecLib.setLiquidationBreakerPriceTolerance(_clip, _pct_bps) [X](#), 745

```

    definition
        ClipperMom.setPriceTolerance(clip, value) X, 258
setRWAilkDebtCeiling
    definition
        DssExecLib.setRWAilkDebtCeiling(_ilk, _ceiling, _price) X, 742
setSha256sum
    call
        DssExecLib.setChangelogSHA256(_SHA256Sum) X, 733
    definition
        ChainLog.setSha256sum(_sha256sum) X a, 710
setStairstepExponentialDecrease
    definition
        DssExecLib.setStairstepExponentialDecrease(_calc, _duration, _pct_bps) X, 746
setStartingPriceMultiplicativeFactor
    call
        DssExecLib.addNewCollateral(co) X, 750
    definition
        DssExecLib.setStartingPriceMultiplicativeFactor(_ilk, _pct_bps) X, 744
setSurplusAuctionAmount
    definition
        DssExecLib.setSurplusAuctionAmount(_amount) X, 738
setSurplusAuctionBidDuration
    definition
        DssExecLib.setSurplusAuctionBidDuration(_duration) X, 738
setSurplusAuctionDuration
    definition
        DssExecLib.setSurplusAuctionDuration(_duration) X, 738
setSurplusBuffer
    definition
        DssExecLib.setSurplusBuffer(_amount) X, 738
settle
    call
        L1DAITeleportGateway.finalizeFlush(targetDomain, daiToFlush) X, 623
        TeleportRouter.settle(targetDomain, batchedDaiToFlush) X, 655
    definition
        TeleportJoin.settle(sourceDomain, batchedDaiToFlush) X, 645
        TeleportRouterMock.settle(targetDomain, batchedDaiToFlush) X, 633
        TeleportRouter.settle(targetDomain, batchedDaiToFlush) X, 655
setUp
    call
        CalleeMakerOtcDai.constructor(otc_, daiJoin_) X, 550
        UniswapV2CalleeDai.constructor(uniRouter02_, daiJoin_) X, 557
        UniswapV2LpTokenCalleeDai.constructor(uniRouter02_, daiJoin_) X, 559
    definition
        CalleeMakerOtcDai.setUp(otc_, daiJoin_) [CalleeMakerOtc], 549
        UniswapV2CalleeDai.setUp(uniRouter02_, daiJoin_) [UniswapV2Callee], 557
        UniswapV2LpTokenCalleeDai.setUp(uniRouter02_, daiJoin_) [UniswapV2Callee(2)], 559
setValue
    call
        DssExecLib.addNewCollateral(co) X, 750
        DssExecLib.decreaseGlobalDebtCeiling(_amount) X, 737
        DssExecLib.decreaseIlkDebtCeiling(_ilk, _amount, _global) X, 742
        DssExecLib.increaseGlobalDebtCeiling(_amount) X, 737
        DssExecLib.increaseIlkDebtCeiling(_ilk, _amount, _global) X, 741
        DssExecLib.setAuctionPermittedDrop(_ilk, _pct_bps) X, 745
        DssExecLib.setAuctionTimeBeforeReset(_ilk, _duration) X, 745
        DssExecLib.setD3MTargetInterestRate(_d3m, _pct_bps) X, 749
        DssExecLib.setDAIReferenceValue(_value) X, 741
        DssExecLib.setDebtAuctionBidDuration(_duration) X, 739
        DssExecLib.setDebtAuctionDAIAmount(_amount) X, 739
        DssExecLib.setDebtAuctionDelay(_duration) X, 739
        DssExecLib.setDebtAuctionDuration(_duration) X, 740
        DssExecLib.setDebtAuctionMKRAmount(_amount) X, 739

```

DssExecLib.setDebtAuctionMKRIncreaseRate(_pct.bps) [X](#), 740
 DssExecLib.setDSR(_rate, _doDrip) [X](#), 737
 DssExecLib.setEmergencyShutdownProcessingTime(_duration) [X](#), 740
 DssExecLib.setExponentialDecrease(_calc, _pct.bps) [X](#), 747
 DssExecLib.setGlobalDebtCeiling(_amount) [X](#), 737
 DssExecLib.setGlobalStabilityFee(_rate) [X](#), 741
 DssExecLib.setIlkDebtCeiling(_ilk, _amount) [X](#), 741
 DssExecLib.setIlkLiquidationPenalty(_ilk, _pct.bps) [X](#), 743
 DssExecLib.setIlkLiquidationRatio(_ilk, _pct.bps) [X](#), 744
 DssExecLib.setIlkMaxLiquidationAmount(_ilk, _amount) [X](#), 744
 DssExecLib.setIlkMinVaultAmount(_ilk, _amount) [X](#), 743
 DssExecLib.setIlkStabilityFee(_ilk, _rate, _doDrip) [X](#), 746
 DssExecLib.setKeeperIncentiveFlatRate(_ilk, _amount) [X](#), 745
 DssExecLib.setKeeperIncentivePercent(_ilk, _pct.bps) [X](#), 745
 DssExecLib.setLinearDecrease(_calc, _duration) [X](#), 746
 DssExecLib.setMaxTotalDAILiquidationAmount(_amount) [X](#), 740
 DssExecLib.setMaxTotalDAILiquidationAmountLEGACY(_amount) [X](#), 740
 DssExecLib.setMinDebtAuctionBidIncrease(_pct.bps) [X](#), 739
 DssExecLib.setMinSurplusAuctionBidIncrease(_pct.bps) [X](#), 738
 DssExecLib.setStairstepExponentialDecrease(_calc, _duration, _pct.bps) [X](#), 746
 DssExecLib.setStartingPriceMultiplicativeFactor(_ilk, _pct.bps) [X](#), 744
 DssExecLib.setSurplusAuctionAmount(_amount) [X](#), 738
 DssExecLib.setSurplusAuctionBidDuration(_duration) [X](#), 738
 DssExecLib.setSurplusAuctionDuration(_duration) [X](#), 738
 DssExecLib.setSurplusBuffer(_amount) [X](#), 738
 definition
 DssExecLib.setValue(_base, _ilk, _what, _amt) [X](#), 736
 DssExecLib.setValue(_base, _what, _amt) [X](#), 736
 setVersion
 call
 ChainLog.constructor() [X](#), 709
 DssExecLib.setChangelogVersion(_version) [X](#), 733
 definition
 ChainLog.setVersion(_version) [X](#) [a](#), 710
 setWard
 call
 GodMode.setWard(base, target, val), 767, 768
 MCD.giveAdminAccess(dss, who), 721
 definition
 GodMode.setWard(base, target, val), 767, 768
 sha256sum
 write
 ChainLog.setSha256sum(_sha256sum) [X](#) [a](#), 710
 definition
 ChainLog.sha256sum: [string](#), 709
 shift
 call
 DssProxyActions.shift(manager, cdpSrc, cdpOrg) [X](#), 534
 definition
 DssCdpManager.shift(cdpSrc, cdpDst) [X](#), 525
 DssProxyActions.shift(manager, cdpSrc, cdpOrg) [X](#), 534
 sig
 write
 DssExec.constructor(_expiration, _spellAction) [X](#), 309
 definition
 DssExec.sig: [bytes](#), 309
 read
 DssExec.cast() [X](#), 310
 DssExec.schedule() [X](#), 309
 signers
 write
 TeleportOracleAuth.addSigners(signers_) [X](#) [a](#), 650
 TeleportOracleAuth.removeSigners(signers_) [X](#) [a](#), 650


```

    TrustedRelay.addSigners(signers_) X, 663
    TrustedRelay.removeSigners(signers_) X, 663
definition
    TeleportOracleAuth.signers: mapping(address => uint256), 649
    TrustedRelay.signers: mapping(address => uint256), 661
read
    TeleportOracleAuth.isValid(signHash, signatures, threshold_), 650
    TrustedRelay.requestMint(teleportGUID, signatures, maxFeePercentage, gasFee,
        expiry, v, r, s), 664
Sin
    write
        Vow.cage() X a, 271
        Vow.fess(tab) X a, 270
        Vow.flog(era) X, 270
    definition
        Vow.Sin: uint256, 269
    read
        Vow.fess(tab) X a, 270
        Vow.flap() X, 271
        Vow.flog(era) X, 270
        Vow.flop() X, 271
        Vow.heal(rad) X, 270
sin
    call
        Vow.cage() X a, 271
        Vow.flap() X, 271
        Vow.flop() X, 271
        Vow.heal(rad) X, 270
    write
        Vat.grab(i, u, v, w, dink, dart) X a, 233
        Vat.heal(rad) X, 233
        Vat.suck(u, v, rad) X a, 234
        Vow.fess(tab) X a, 270
        Vow.flog(era) X, 270
    definition
        Vat.sin: mapping(address => uint256), 229
        Vow.sin: mapping(uint256 => uint256), 269
    read
        Vat.grab(i, u, v, w, dink, dart) X a, 233
        Vat.heal(rad) X, 233
        Vat.suck(u, v, rad) X a, 234
        Vow.fess(tab) X a, 270
        Vow.flog(era) X, 270
skim
    call
        D3MHub.exec(ilk) X, 324
        DssProxyActionsEnd._free(manager, end, cdp), 543
    definition
        End.skim(ilk, urn) X, 300
skip
    definition
        End.skip(ilk, id) X, 299
skipWhenForking
    definition
        D3MCoreDeployScript.modifier skipWhenForking() [StdCheatsSafe], 334
        D3MCoreInitScript.modifier skipWhenForking() [StdCheatsSafe], 362
        D3MDeployScript.modifier skipWhenForking() [StdCheatsSafe], 388
        D3MInitScript.modifier skipWhenForking() [StdCheatsSafe], 409
skipWhenNotForking
    definition
        D3MCoreDeployScript.modifier skipWhenNotForking() [StdCheatsSafe], 335
        D3MCoreInitScript.modifier skipWhenNotForking() [StdCheatsSafe], 363
        D3MDeployScript.modifier skipWhenNotForking() [StdCheatsSafe], 389

```

D3MInitScript.modifier skipWhenNotForking() [StdCheatsSafe], 409

slip

call

- AuthGemJoin5.exit(usr, amt) X, 504
- AuthGemJoin5.join(urn, amt, msgSender) X a, 504
- AuthGemJoin8.exit(usr, amt) X, 506
- AuthGemJoin8.join(urn, amt, msgSender) X a, 506
- AuthGemJoin.exit(guy, wad) X, 508
- AuthGemJoin.exit(usr, wad) X, 448
- AuthGemJoin.join(urn, wad, msgSender) X a, 508
- AuthGemJoin.join(usr, wad) X a, 448
- D3MHub._exec(ilk, _pool, Art, lineWad), 322
- D3MHub.exit(ilk, usr, wad) X, 325
- D3MHub._wipe(ilk, _pool, urn), 322
- GemJoin2.exit(usr, wad) X, 428
- GemJoin2.join(usr, wad) X, 428
- GemJoin3.exit(usr, amt) X, 430
- GemJoin3.join(usr, amt) X, 430
- GemJoin4.exit(usr, wad) X, 434
- GemJoin4.join(usr, wad) X, 434
- GemJoin5.exit(usr, amt) X, 436
- GemJoin5.join(usr, amt) X, 436
- GemJoin6.exit(usr, wad) X, 438
- GemJoin6.join(usr, wad) X, 438
- GemJoin7.exit(usr, amt) X, 440
- GemJoin7.join(usr, amt) X, 440
- GemJoin8.exit(usr, amt) X, 442
- GemJoin8.join(usr, amt) X, 442
- GemJoin9.exit(usr, wad) X, 445
- GemJoin9._join(usr), 444
- GemJoin.exit(usr, wad) X, 424
- GemJoin.join(usr, wad) X, 424
- ManagedGemJoin.exit(urn, usr, amt) X a, 450
- ManagedGemJoin.join(usr, amt) X a, 450
- TeleportJoin._mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643
- TeleportJoin.settle(sourceDomain, batchedDaiToFlush) X, 645

definition

- Vat.slip(ilk, usr, wad) X a, 231

slot

write

- Median.drop(a) X a, 283
- Median.lift(a) X a, 283

definition

- Median.slot: mapping(uint8 => address), 281

read

- Median.lift(a) X a, 283

snip

definition

- End.snip(ilk, id) X, 299

SN_PRIME

definition

- L1DAIBridge.SN_PRIME: uint256, 617

read

- L1DAIBridge.deposit(amount, l2Recipient) X, 619

_spendAllowance

call

- DAIMock.transferFrom(from, to, amount) [ERC20] X, 628

definition

- DAIMock._spendAllowance(owner, spender, amount) [ERC20], 631

splitSignature

call

- TeleportOracleAuth.isValid(signHash, signatures, threshold_), 650

definition

TeleportOracleAuth.splitSignature(signatures, index), 651

spot

- write
 - End.file(what, data) X a, 298
 - IlkRegistry.constructor(vat_, dog_, cat_, spot_) X, 668
 - IlkRegistry.file(what, data) X a, 669
 - Vat.file(ilk, what, data) X a, 231
- definition
 - End.spot: SpotLike, 295
 - IlkRegistry.spot: SpotLike, 667
- read
 - End.cage() X a, 299
 - End.cage(ilk) X, 299
 - IlkRegistry.add(adapter) X, 668
 - IlkRegistry.constructor(vat_, dog_, cat_, spot_) X, 668
 - IlkRegistry.update(ilk) X, 673

Spotter

- definition
 - contract Spotter, 227

spotter

- call
 - DssExecLib.addCollateralBase(_ilk, _gem, _join, _clip, _calc, _pip) X, 749
 - DssExecLib.addNewCollateral(co) X, 750
 - DssExecLib.setDAIReferenceValue(_value) X, 741
 - DssExecLib.setIlkLiquidationRatio(_ilk, _pct_bps) X, 744
 - DssExecLib.updateCollateralPrice(_ilk) X, 736
- write
 - Clipper.constructor(vat_, spotter_, dog_, ilk_) X, 243
 - Clipper.file(what, data) X a, 243
 - ClipperMom.constructor(spotter_) X, 257
- definition
 - ClipperMom.spotter: SpotterLike, 257
 - Clipper.spotter: SpotterLike, 241
 - DssExecLib.spotter(), 731
- read
 - Clipper.getFeedPrice(), 244
 - ClipperMom.getPrices(clip), 258

src

- call
 - DssExecLib.whitelistOracleMedians(_oracle) X, 747
- write
 - OSM.constructor(src_) X, 286
 - OSM.change(src_) X a, 287
- definition
 - OSM.src: address, 285
- read
 - OSM.poke() X, 287

srcs

- write
 - Cure.drop(src) X a, 293
 - Cure.lift(src) X a, 293
- definition
 - Cure.srcs: address[], 291
- read
 - Cure.drop(src) X a, 293
 - Cure.lift(src) X a, 293
 - Cure.list(), 292
 - Cure.tCount(), 292
 - Cure.tell(), 292

stableDebt

- call
 - D3MInit.initAave(dss, d3m, cfg, aaveCfg), 726
 - D3MInitScript.run() X, 420

write
 D3MAavePlan.constructor(dai_, pool_) X, 311
 D3MAavePool.constructor(ilk_, hub_, dai_, pool_) X, 349
 definition
 D3MAavePlan.stableDebt: TokenLike, 311
 D3MAavePool.stableDebt: ATokenLike, 349
 read
 D3MAavePlan.active(), 314
 D3MAavePlan.getTargetAssets(currentAssets), 313
 StairstepExponentialDecrease
 definition
 contract StairstepExponentialDecrease, 237
 starkNet
 write
 L1DAIBridge.constructor(_starkNet, _dai, _l2Dai, _escrow, _l2DaiBridge) X, 618
 L1DAITeleportGateway.constructor(_starkNet, _l1Token, _l2TeleportGateway,
 _l1Escrow, _l1TeleportRouter) X, 623
 L1GovernanceRelay.constructor(_starkNet, _l2GovernanceRelay) X, 621
 definition
 L1DAIBridge.starkNet: address, 617
 L1DAITeleportGateway.starkNet: address, 623
 L1GovernanceRelay.starkNet: address, 621
 read
 L1DAIBridge.cancelDeposit(amount, l2Recipient, l1Recipient, nonce) X, 620
 L1DAIBridge.deposit(amount, l2Recipient) X, 619
 L1DAIBridge.forceWithdrawal(amount, l2Sender) X, 620
 L1DAIBridge.startDepositCancellation(amount, l2Recipient, nonce) X, 620
 L1DAIBridge.withdraw(amount, l1Recipient) X, 619
 L1DAITeleportGateway.finalizeFlush(targetDomain, daiToFlush) X, 623
 L1DAITeleportGateway.finalizeRegisterTeleport(teleport) X, 623
 L1GovernanceRelay.relay(spell) X a, 621
 start
 definition
 OSM.start() X a, 286
 startBroadcast
 call
 D3MCoreDeployScript.run() X, 346
 D3MCoreInitScript.run() X, 374
 D3MDeployScript.run() X, 400
 D3MInitScript.run() X, 420
 startDepositCancellation
 definition
 L1DAIBridge.startDepositCancellation(amount, l2Recipient, nonce) X, 620
 startL1ToL2MessageCancellation
 call
 L1DAIBridge.startDepositCancellation(amount, l2Recipient, nonce) X, 620
 status
 call
 Clipper.getStatus(id), 248
 Clipper.redo(id, kpr) X, 245
 Clipper.take(id, amt, max, who, data) X, 246
 definition
 Clipper.status(tic, top), 248
 stdstore
 definition
 D3MCoreDeployScript.stdstore [CommonBase] : StdStorage, 329
 D3MCoreInitScript.stdstore [CommonBase] : StdStorage, 357
 D3MDeployScript.stdstore [CommonBase] : StdStorage, 383
 D3MInitScript.stdstore [CommonBase] : StdStorage, 403
 step
 write
 StairstepExponentialDecrease.file(what, data) X a, 237
 definition

```

    OSM.step(ts) X a, 287
    StairstepExponentialDecrease.step: uint256, 237
read
    StairstepExponentialDecrease.price(top, dur), 238
stETH
    call
        WstETHCurveUniv3Callee.clipperCall(sender, owe, slice, data) X, 564
stop
    definition
        OSM.stop() X a, 286
stopBroadcast
    call
        D3MCoreDeployScript.run() X, 346
        D3MCoreInitScript.run() X, 374
        D3MDeployScript.run() X, 400
        D3MInitScript.run() X, 420
stoppable
    call
        OSM.poke() X, 287
    definition
        OSM.modifier stoppable(), 286
stopped
    call
        ClipperMom.tripBreaker(clip) X, 259
write
    Clipper.file(what, data) X a, 243
    OSM.start() X a, 286
    OSM.stop() X a, 286
    OSM.void() X a, 287
definition
    Clipper.stopped: uint256, 241
    OSM.stopped: uint256, 285
read
    Clipper.modifier isStopped(level), 242
    OSM.modifier stoppable(), 286
store
    call
        GodMode.setBalance(token, who, amount), 768
        GodMode.setWard(base, target, val), 767
stringToBytes32
    call
        D3MDeployScript.run() X, 400
        D3MInit._init(dss, d3m, cfg, gem), 725
        D3MInitScript.run() X, 420
definition
    ScriptTools.stringToBytes32(source), 715
sub
    call
        AAVE.transferFrom(src, dst, wad) X, 452
        BAL.transferFrom(src, dst, wad) X, 454
        BAT.transferFrom(src, dst, wad) X, 456
        CalleeMakerOtcDai.divup(x, y) [CalleeMakerOtc], 549
        CalleeMakerOtcDai._fromWad(gemJoin, wad) [CalleeMakerOtc], 550
        Clipper.status(tic, top), 248
        COMP.transferFrom(src, dst, wad) X, 458
        Cure.load(src) X, 293
        CurveLpTokenUniv3Callee._divup(x, y), 551
        CurveLpTokenUniv3Callee._fromWad(gemJoin, wad), 552
        Dai.burn(usr, wad) X, 225
        Dai.transferFrom(src, dst, wad) X, 224
        DGD.transferFrom(src, dst, wad) X, 460
        Dog.digs(ilk, rad) X a, 255
        DssAutoLine.exec(_ilk) X, 707

```

DssCdpManager.give(cdp, dst) [X](#), 523
DssProxyActions._getDrawDart(vat, jug, urn, ilk, wad), 530
DssProxyActions._getWipeAllWad(vat, usr, urn, ilk), 530
DssPsm.sellGem(usr, gemAmt) [X](#), 510
DssVestMintable.accrued(_time, _bgn, _fin, _tot) [DssVest], 680
DssVestMintable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] [X a](#), 678
DssVestMintable.unpaid(_time, _bgn, _clf, _fin, _tot, _rx) [DssVest], 680
DssVestSuckable.accrued(_time, _bgn, _fin, _tot) [DssVest], 690
DssVestSuckable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] [X a](#), 688
DssVestSuckable.unpaid(_time, _bgn, _clf, _fin, _tot, _rx) [DssVest], 690
DssVestTransferrable.accrued(_time, _bgn, _fin, _tot) [DssVest], 700
DssVestTransferrable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] [X a](#), 698
DssVestTransferrable.unpaid(_time, _bgn, _clf, _fin, _tot, _rx) [DssVest], 700
End.flow(ilk) [X](#), 300
End.skim(ilk, urn) [X](#), 300
End.thaw() [X](#), 300
Flapper.deal(id) [X](#), 264
GemJoin7.join(usr, amt) [X](#), 440
GemJoin9.exit(usr, wad) [X](#), 445
GemJoin9._join(usr), 444
GNT.transfer(dst, wad) [X](#), 461
GUSD.transferFrom(src, dst, wad) [X](#), 464
KNC.transferFrom(src, dst, wad) [X](#), 466
LINK.transferFrom(src, dst, wad) [X](#), 468
LRC.transferFrom(src, dst, wad) [X](#), 470
MANA.transferFrom(src, dst, wad) [X](#), 472
MATIC.transferFrom(src, dst, wad) [X](#), 474
OMG.transferFrom(src, dst, wad) [X](#), 476
PAXG.transferFrom(src, dst, wad) [X](#), 478
PAXUSD.transferFrom(src, dst, wad) [X](#), 482
Pot.drip() [X](#), 279
Pot.exit(wad) [X](#), 279
PSMCallee._divup(x, y), 555
PSMCallee._fromWad(gemJoin, wad), 555
RENBTC.transferFrom(src, dst, wad) [X](#), 484
REP.transferFrom(src, dst, wad) [X](#), 486
TUSD.transferFrom(src, dst, wad) [X](#), 488
UniswapV2CalleeDai.divup(x, y) [UniswapV2Callee], 557
UniswapV2CalleeDai._fromWad(gemJoin, wad) [UniswapV2Callee], 557
UniswapV2LpTokenCalleeDai.divup(x, y) [UniswapV2Callee(2)], 559
UniswapV2LpTokenCalleeDai._fromWad(gemJoin, wad) [UniswapV2Callee(2)], 559
UniswapV3Callee._divup(x, y), 567
UniswapV3Callee._fromWad(gemJoin, wad), 567
UNI.transferFrom(src, dst, wad) [X](#), 490
USDC.transferFrom(src, dst, wad) [X](#), 492
USDT.transfer(_to, _value) [X](#), 493
USDT.transferFrom(_from, _to, _value) [X](#), 494
Vat.flux(ilk, src, dst, wad) [X](#), 231
Vat.fork(ilk, src, dst, dink, dart) [X](#), 233
Vat.frob(i, u, v, w, dink, dart) [X](#), 232
Vat.grab(i, u, v, w, dink, dart) [X a](#), 233
Vat.heal(rad) [X](#), 233
Vat.move(src, dst, rad) [X](#), 232
Vow.flap() [X](#), 271
Vow.flog(era) [X](#), 270
Vow.flop() [X](#), 271
Vow.heal(rad) [X](#), 270
Vow.kiss(rad) [X](#), 271
WBTC.transferFrom(src, dst, wad) [X](#), 496
WstETHCurveUniv3Callee._divup(x, y), 563
WstETHCurveUniv3Callee._fromWad(gemJoin, wad), 563
WSTETH.transferFrom(src, dst, wad) [X](#), 498
YFI.transferFrom(src, dst, wad) [X](#), 500

ZRX.transferFrom(src, dst, wad) [X](#), 502

definition

AAVE.sub(x, y), 451
 BAL.sub(x, y), 453
 BAT.sub(x, y), 455
 CalleeMakerOtcDai.sub(x, y) [CalleeMakerOtc], 549
 Clipper.sub(x, y), 243
 COMP.sub(x, y), 457
 Cure.sub(x, y), 291
 CurveLpTokenUniv3Callee.sub(x, y), 551
 Dai.sub(x, y), 224, 588, 614
 DGD.sub(x, y), 459
 Dog.sub(x, y), 252
 DssAutoLine.sub(x, y), 706
 DssCdpManager.sub(x, y), 522
 DssProxyActions.sub(x, y), 529
 DssPsm.sub(x, y), 510
 DssVestMintable.sub(x, y) [DssVest], 678
 DssVestSuckable.sub(x, y) [DssVest], 688
 DssVestTransferrable.sub(x, y) [DssVest], 698
 End.sub(x, y), 298
 Flapper.sub(x, y), 262
 GemJoin7.sub(x, y), 440
 GemJoin9.sub(x, y), 444
 GNT.sub(x, y), 461
 GUSD.sub(x, y), 463
 KNC.sub(x, y), 465
 LINK.sub(x, y), 467
 LRC.sub(x, y), 469
 MANA.sub(x, y), 471
 MATIC.sub(x, y), 473
 OMG.sub(x, y), 475
 PAXG.sub(x, y), 477
 PAXUSD.sub(x, y), 481
 Pot.sub(x, y), 278
 PSMCallee.sub(x, y), 555
 RENBTC.sub(x, y), 483
 REP.sub(x, y), 485
 SafeMath.sub(a, b), 753
 TUSD.sub(x, y), 487
 UNI.sub(x, y), 489
 UniswapV2CalleeDai.sub(x, y) [UniswapV2Callee], 557
 UniswapV2LpTokenCalleeDai.sub(x, y) [UniswapV2Callee(2)], 559
 UniswapV3Callee.sub(x, y), 567
 USDC.sub(x, y), 491
 Vat.sub(x, y), 230, 231
 Vow.sub(x, y), 270
 WBTC.sub(x, y), 495
 WstETHCurveUniv3Callee.sub(x, y), 563
 WSTETH.sub(x, y), 497
 YFI.sub(x, y), 499
 ZRX.sub(x, y), 501

suck

call

Clipper.kick(tab, lot, usr, kpr) [X](#) [a](#), 244
 Clipper.redo(id, kpr) [X](#), 245
 D3MHub.exec(ilk, _pool, Art, lineWad), 322
 D3MHub.uncull(ilk) [X](#), 326
 DssExecLib.sendPaymentFromSurplusBuffer(_target, _amount) [X](#), 751
 DssVestSuckable.pay(_guy, _amt), 693
 End.skip(ilk, id) [X](#), 299
 End.snip(ilk, id) [X](#), 299
 Flopper.yank(id) [X](#), 267

Pot.drip() [X](#), 279
 definition
 Vat.suck(u, v, rad) [X](#) [a](#), 234
 Sum
 write
 ESM.join(wad) [X](#), 305
 definition
 ESM.Sum: [uint256](#), 303
 read
 ESM.denyProxy(target) [X](#), 305
 ESM.fire() [X](#), 304
 ESM.join(wad) [X](#), 305
 sum
 write
 ESM.join(wad) [X](#), 305
 definition
 ESM.sum: [mapping](#)([address](#) => [uint256](#)), 303
 read
 ESM.join(wad) [X](#), 305
 sump
 write
 Vow.file(what, [data](#)) [X](#) [a](#), 270
 definition
 Vow.sump: [uint256](#), 269
 read
 Vow.flop() [X](#), 271
 _supply
 write
 AAVE.constructor(supply) [X](#), 451
 BAL.constructor(supply) [X](#), 453
 BAT.constructor(supply) [X](#), 455
 COMP.constructor(supply) [X](#), 457
 DGD.constructor(supply) [X](#), 459
 GNT.constructor(supply) [X](#), 461
 GUSD.constructor(supply) [X](#), 463
 KNC.constructor(supply) [X](#), 465
 LINK.constructor(supply) [X](#), 467
 LRC.constructor(supply) [X](#), 469
 MANA.constructor(supply) [X](#), 471
 MATIC.constructor(supply) [X](#), 473
 OMG.constructor(supply) [X](#), 475
 PAXG.constructor(supply, fee) [X](#), 478
 PAXUSD.constructor(supply) [X](#), 481
 RENBTC.constructor(supply) [X](#), 483
 REP.constructor(supply) [X](#), 485
 TUSD.constructor(supply) [X](#), 487
 UNI.constructor(supply) [X](#), 489
 USDC.constructor(supply) [X](#), 491
 WBTC.constructor(supply) [X](#), 495
 WSTETH.constructor(supply) [X](#), 497
 YFI.constructor(supply) [X](#), 499
 ZRX.constructor(supply) [X](#), 501
 definition
 AAVE._supply: [uint256](#), 451
 BAL._supply: [uint256](#), 453
 BAT._supply: [uint256](#), 455
 COMP._supply: [uint256](#), 457
 DGD._supply: [uint256](#), 459
 GNT._supply: [uint256](#), 461
 GUSD._supply: [uint256](#), 463
 KNC._supply: [uint256](#), 465
 LINK._supply: [uint256](#), 467
 LRC._supply: [uint256](#), 469


```

    MANA._supply: uint256, 471
    MATIC._supply: uint256, 473
    OMG._supply: uint256, 475
    PAXG._supply: uint256, 477
    PAXUSD._supply: uint256, 481
    RENBTC._supply: uint256, 483
    REP._supply: uint256, 485
    TUSD._supply: uint256, 487
    UNI._supply: uint256, 489
    USDC._supply: uint256, 491
    WBTC._supply: uint256, 495
    WSTETH._supply: uint256, 497
    YFI._supply: uint256, 499
    ZRX._supply: uint256, 501
read
    AAVE.totalSupply(), 451
    BAL.totalSupply(), 453
    BAT.totalSupply(), 455
    COMP.totalSupply(), 457
    DGD.totalSupply(), 459
    GNT.totalSupply(), 461
    GUSD.totalSupply(), 463
    KNC.totalSupply(), 465
    LINK.totalSupply(), 467
    LRC.totalSupply(), 469
    MANA.totalSupply(), 471
    MATIC.totalSupply(), 473
    OMG.totalSupply(), 475
    PAXG.totalSupply(), 478
    PAXUSD.totalSupply(), 481
    RENBTC.totalSupply(), 483
    REP.totalSupply(), 485
    TUSD.totalSupply(), 487
    UNI.totalSupply(), 489
    USDC.totalSupply(), 491
    WBTC.totalSupply(), 495
    WSTETH.totalSupply(), 497
    YFI.totalSupply(), 499
    ZRX.totalSupply(), 501
swapExactTokensForTokens
    call
        UniswapV2CalleeDai.clipperCall(sender, daiAmt, gemAmt, data) X, 558
        UniswapV2LpTokenCalleeDai.swapGemForDai(token, path, to), 560
swapGemForDai
    call
        UniswapV2LpTokenCalleeDai.clipperCall(sender, daiAmt, gemAmt, data) X, 560
    definition
        UniswapV2LpTokenCalleeDai.swapGemForDai(token, path, to), 560
switchOwner
    call
        D3MDeploy.deployAave(deployer, owner, ilk, vat, hub, dai, lendingPool), 723
        D3MDeploy.deployCompound(deployer, owner, ilk, vat, hub, cdai), 723
        D3MDeploy.deployCore(deployer, owner, daiJoin), 723
    definition
        ScriptTools.switchOwner(base, deployer, newOwner), 716
symbol
    call
        D3MInit._init(dss, d3m, cfg, gem), 725
        GemInfo.symbol(token), 665
        IlkRegistry.add(adapter) X, 668
    write
        DAIMock.constructor(name_, symbol_) [ERC20] X, 626
        IlkRegistry.file(ilk, what, data) X a, 670

```

definition

AAVE.symbol: [string](#), 451
BAL.symbol: [string](#), 453
BAT.symbol: [string](#), 455
COMP.symbol: [string](#), 457
DAIMock.symbol [ERC20] : [string](#), 625
DAIMock.symbol() [ERC20], 626
Dai.symbol: [string](#), 223, 587, 613
DGD.symbol: [string](#), 459
GemInfo.symbol(token), 665
GNT.symbol: [string](#), 461
GUSD.symbol: [string](#), 463
IlkRegistry.symbol(ilk), 672
KNC.symbol: [string](#), 465
LINK.symbol: [string](#), 467
LRC.symbol: [string](#), 469
MANA.symbol: [string](#), 471
MATIC.symbol: [string](#), 473
OMG.symbol: [string](#), 475
PAXG.symbol: [string](#), 477
PAXUSD.symbol: [string](#), 481
RENBTC.symbol: [string](#), 483
REP.symbol: [string](#), 485
TUSD.symbol: [string](#), 487
UNI.symbol: [string](#), 489
USDC.symbol: [string](#), 491
USDT.symbol: [string](#), 493
WBTC.symbol: [string](#), 495
WETH9.symbol: [string](#), 517
WSTETH.symbol: [string](#), 497
YFI.symbol: [string](#), 499
ZRX.symbol: [string](#), 501

read

DAIMock.symbol() [ERC20], 626
IlkRegistry.symbol(ilk), 672

T

tab

write

Clipper.kick(tab, lot, usr, kpr) **X** **a**, 244
 Clipper.take(id, amt, max, who, **data**) **X**, 246

read

Clipper.getStatus(id), 248
 Clipper.redo(id, kpr) **X**, 245
 Clipper.take(id, amt, max, who, **data**) **X**, 246
 Clipper.yank(id) **X** **a**, 249

tack

call

D3MInit.initAave(dss, d3m, cfg, aaveCfg), 726
 D3MInit.initCompound(dss, d3m, cfg, compoundCfg), 727
 D3MInitScript.run() **X**, 420

write

D3MAavePlan.constructor(dai_, pool_) **X**, 311
 D3MAavePlan.file(what, **data**) **X** **a**, 312

definition

D3MAavePlan.tack: InterestRateStrategyLike, 311

read

D3MAavePlan.active(), 314
 D3MAavePlan._calculateTargetSupply(targetInterestRate, totalDebt), 313

tacks

write

D3MCompoundPlan.constructor(cDai_) **X**, 315
 D3MCompoundPlan.file(what, addr, **data**) **X** **a**, 316

definition

D3MCompoundPlan.tacks: **mapping**(address => uint256), 315

read

D3MCompoundPlan.active(), 317
 D3MCompoundPlan._calculateTargetSupply(targetInterestRate, borrows), 316

tag

write

DssExec.constructor(_expiration, _spellAction) **X**, 309
 End.cage(ilk) **X**, 299

definition

DssExec.tag: **bytes32**, 309
 End.tag: **mapping**(bytes32 => uint256), 295

read

DssExec.cast() **X**, 310
 DssExec.schedule() **X**, 309
 End.cage(ilk) **X**, 299
 End.flow(ilk) **X**, 300
 End.skim(ilk, urn) **X**, 300
 End.skip(ilk, id) **X**, 299
 End.snip(ilk, id) **X**, 299

tail

write

Clipper.file(what, **data**) **X** **a**, 243

definition

Clipper.tail: **uint256**, 241

read

Clipper.status(tic, top), 248

take

definition

Clipper.take(id, amt, max, who, **data**) **X**, 246

tau

write

D3MHub.file(ilk, what, **data**) **X** **a**, 321
 Flapper.file(what, **data**) **X** **a**, 263
 Flopper.file(what, **data**) **X** **a**, 266
 LinearDecrease.file(what, **data**) **X** **a**, 236

```

definition
    D3MHub.tau(ilk), 327
    Flapper.tau: uint48, 261
    Flopper.tau: uint48, 265
    LinearDecrease.tau: uint256, 235
read
    D3MHub.cage(ilk) X a, 325
    D3MHub.tau(ilk), 327
    Flapper.kick(lot, bid) X a, 263
    Flapper.tick(id) X, 263
    Flopper.kick(gal, lot, bid) X a, 266
    Flopper.tick(id) X, 267
    LinearDecrease.price(top, dur), 236
tCount
    definition
        Cure.tCount(), 292
TeleportConstantFee
    definition
        contract TeleportConstantFee, 639
TeleportJoin
    definition
        contract TeleportJoin, 641
teleportJoin
    call
        BasicRelay.constructor(_oracleAuth, _daiJoin) X, 657
        TrustedRelay.constructor(_oracleAuth, _daiJoin, _ethPriceOracle) X, 661
    write
        BasicRelay.constructor(_oracleAuth, _daiJoin) X, 657
        TeleportOracleAuth.constructor(teleportJoin_) X, 649
        TrustedRelay.constructor(_oracleAuth, _daiJoin, _ethPriceOracle) X, 661
    definition
        BasicRelay.teleportJoin: TeleportJoinLike, 657
        TeleportOracleAuth.teleportJoin: TeleportJoinLike, 649
        TrustedRelay.teleportJoin: TeleportJoinLike, 661
    read
        TeleportOracleAuth.requestMint(teleportGUID, signatures, maxFeePercentage,
            operatorFee) X, 650
TeleportLinearFee
    definition
        contract TeleportLinearFee, 647
TeleportOracleAuth
    definition
        contract TeleportOracleAuth, 649
TeleportRouter
    definition
        contract TeleportRouter, 653
TeleportRouterMock
    definition
        contract TeleportRouterMock, 633
teleports
    write
        TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643
        TeleportJoin.requestMint(teleportGUID, maxFeePercentage, operatorFee) X a, 644
    definition
        TeleportJoin.teleports: mapping(bytes32 => TeleportJoin.TeleportStatus), 641
    read
        TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643
        TeleportJoin.requestMint(teleportGUID, maxFeePercentage, operatorFee) X a, 644
TeleportStatus
    definition
        struct TeleportJoin.TeleportStatus, 641
tell
    call

```

```

    End.thaw() X, 300
definition
    Cure.tell(), 292
tend
definition
    Flapper.tend(id, lot, bid) X, 263
thaw
definition
    End.thaw() X, 300
THOUSAND
definition
    DssExecLib.THOUSAND: uint256, 729
threshold
write
    TeleportOracleAuth.file(what, data) X a, 649
definition
    TeleportOracleAuth.threshold: uint256, 649
read
    TeleportOracleAuth.requestMint(teleportGUID, signatures, maxFeePercentage,
        operatorFee) X, 650
tic
write
    Clipper.kick(tab, lot, usr, kpr) X a, 244
    Clipper.redo(id, kpr) X, 245
    D3MHub.cage(ilk) X a, 325
    Flapper.tend(id, lot, bid) X, 263
    Flopper.dent(id, lot, bid) X, 267
definition
    D3MHub.tic(ilk), 327
read
    Clipper.getStatus(id), 248
    Clipper.redo(id, kpr) X, 245
    Clipper.take(id, amt, max, who, data) X, 246
    D3MHub.cage(ilk) X a, 325
    D3MHub.cull(ilk) X, 326
    D3MHub.exec(ilk, _pool, Art, lineWad), 322
    D3MHub.file(ilk, what, data) X a, 321
    D3MHub.tic(ilk), 327
    Flapper.deal(id) X, 264
    Flapper.tend(id, lot, bid) X, 263
    Flapper.tick(id) X, 263
    Flopper.deal(id) X, 267
    Flopper.dent(id, lot, bid) X, 267
    Flopper.tick(id) X, 267
tick
call
    DssExecLib.linearInterpolation(name, _target, _ilk, _what, _startTime, _start,
        _end, _duration) X, 752
    DssExecLib.linearInterpolation(name, _target, _what, _startTime, _start, _end,
        _duration) X, 751
definition
    Flapper.tick(id) X, 263
    Flopper.tick(id) X, 267
tin
write
    DssPsm.file(what, data) X a, 510
definition
    DssPsm.tin: uint256, 509
read
    DssPsm.sellGem(usr, gemAmt) X, 510
tip
write
    Clipper.file(what, data) X a, 243

```

definition
 Clipper.tip: uint192, 241
 read
 Clipper.kick(tab, lot, usr, kpr) X a, 244
 Clipper.redo(id, kpr) X, 245
 to18ConversionFactor
 write
 DssPsm.constructor(gemJoin_, daiJoin_, vow_) X, 509
 definition
 DssPsm.to18ConversionFactor: uint256, 509
 read
 DssPsm.buyGem(usr, gemAmt) X, 511
 DssPsm.sellGem(usr, gemAmt) X, 510
 toInt
 call
 DssCdpManager.enter(src, cdp) X, 525
 DssCdpManager.quit(cdp, dst) X, 524
 DssCdpManager.shift(cdpSrc, cdpDst) X, 525
 DssProxyActions.freeETH(manager, ethJoin, cdp, wad) X, 535
 DssProxyActions.freeGem(manager, gemJoin, cdp, amt) X, 535
 DssProxyActions._getDrawDart(vat, jug, urn, ilk, wad), 530
 DssProxyActions._getWipeDart(vat, dai, urn, ilk), 530
 DssProxyActions.lockETH(manager, ethJoin, cdp) X, 534
 DssProxyActions.lockETHAndDraw(manager, jug, ethJoin, daiJoin, cdp, wadD) X, 538
 DssProxyActions.lockGem(manager, gemJoin, cdp, amt, transferFrom) X, 535
 DssProxyActions.lockGemAndDraw(manager, jug, gemJoin, daiJoin, cdp, amtC, wadD, transferFrom) X, 539
 DssProxyActions.wipeAllAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC) X, 540
 DssProxyActions.wipeAllAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC) X, 541
 DssProxyActions.wipeAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC, wadD) X, 540
 DssProxyActions.wipeAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC, wadD) X, 541
 definition
 DssCdpManager.toInt(x), 522
 DssProxyActions.toInt(x), 529
 token
 write
 L1EscrowMom.constructor(escrow_, token_) X, 637
 definition
 L1EscrowMom.token: address, 637
 read
 L1EscrowMom.refuse(spender) X a, 638
 token0
 call
 UniswapV2LpTokenCalleeDai.clipperCall(sender, daiAmt, gemAmt, data) X, 560
 token1
 call
 UniswapV2LpTokenCalleeDai.clipperCall(sender, daiAmt, gemAmt, data) X, 560
 toL2String
 call
 L1DAITeleportGateway.finalizeFlush(targetDomain, daiToFlush) X, 623
 L1DAITeleportGateway.finalizeRegisterTeleport(teleport) X, 623
 definition
 L1DAITeleportGateway.toL2String(str), 624
 tolerance
 write
 ClipperMom.setPriceTolerance(clip, value) X, 258
 definition
 ClipperMom.tolerance: mapping(address => uint256), 257
 read
 ClipperMom.tripBreaker(clip) X, 259
 toll
 call
 Median.peak(), 282

- Median.read(), 282
- OSM.peek(), 287
- OSM.peep(), 288
- OSM.read(), 288
- TrustedRelay.addSigners(signers_) X, 663
- TrustedRelay.removeSigners(signers_) X, 663
- definition
 - Median.modifier toll(), 282
 - OSM.modifier toll(), 286
 - TrustedRelay.modifier toll(), 661
- top
 - write
 - Clipper.kick(tab, lot, usr, kpr) X a, 244
 - Clipper.redo(id, kpr) X, 245
 - read
 - Clipper.getStatus(id), 248
 - Clipper.redo(id, kpr) X, 245
 - Clipper.take(id, amt, max, who, data) X, 246
- topics
 - write
 - D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341
 - D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369
 - D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395
 - D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415
 - read
 - D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341
 - D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369
 - D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395
 - D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415
- toRad
 - call
 - DssProxyActions.draw(manager, jug, daiJoin, cdp, wad) X, 536
 - DssProxyActions.lockETHAndDraw(manager, jug, ethJoin, daiJoin, cdp, wadD) X, 538
 - DssProxyActions.lockGemAndDraw(manager, jug, gemJoin, daiJoin, cdp, amtC, wadD, transferFrom) X, 539
 - definition
 - DssProxyActions.toRad(wad), 529
- toSplitUint
 - call
 - L1DAIBridge.cancelDeposit(amount, l2Recipient, l1Recipient, nonce) X, 620
 - L1DAIBridge.deposit(amount, l2Recipient) X, 619
 - L1DAIBridge.forceWithdrawal(amount, l2Sender) X, 620
 - L1DAIBridge.startDepositCancellation(amount, l2Recipient, nonce) X, 620
 - L1DAIBridge.withdraw(amount, l1Recipient) X, 619
 - L1DAITeleportGateway.finalizeFlush(targetDomain, daiToFlush) X, 623
 - definition
 - L1DAIBridge.toSplitUint(value), 619
 - L1DAITeleportGateway.toSplitUint(value), 624
- toString
 - call
 - D3MCoreDeployScript.bound(x, min, max) [StdUtils], 336
 - D3MCoreDeployScript.getChain(chainId) [StdChains], 343
 - D3MCoreDeployScript.readReceipt(path, index) [StdCheatsSafe], 340
 - D3MCoreDeployScript.readTx1559(path, index) [StdCheatsSafe], 340
 - D3MCoreDeployScript.setChain(chainAlias, chain) [StdChains], 344
 - D3MCoreInitScript.bound(x, min, max) [StdUtils], 364
 - D3MCoreInitScript.getChain(chainId) [StdChains], 371
 - D3MCoreInitScript.readReceipt(path, index) [StdCheatsSafe], 368
 - D3MCoreInitScript.readTx1559(path, index) [StdCheatsSafe], 368
 - D3MCoreInitScript.setChain(chainAlias, chain) [StdChains], 372
 - D3MDeployScript.bound(x, min, max) [StdUtils], 390
 - D3MDeployScript.getChain(chainId) [StdChains], 397
 - D3MDeployScript.readReceipt(path, index) [StdCheatsSafe], 394

D3MDeployScript.readTx1559(path, index) [StdCheatsSafe], 394
 D3MDeployScript.setChain(chainAlias, chain) [StdChains], 398
 D3MInitScript.bound(x, min, max) [StdUtils], 410
 D3MInitScript.getChain(chainId) [StdChains], 418
 D3MInitScript.readReceipt(path, index) [StdCheatsSafe], 414
 D3MInitScript.readTx1559(path, index) [StdCheatsSafe], 414
 D3MInitScript.setChain(chainAlias, chain) [StdChains], 418
 ScriptTools.exportContract(name, label, addr), 715
 ScriptTools.readInput(name), 713
 ScriptTools.readOutput(name), 714
 ScriptTools.readOutput(name, **timestamp**), 713

tot

write

DssVestMintable._yank(_id, _end) [DssVest], 682
 DssVestSuckable._yank(_id, _end) [DssVest], 691
 DssVestTransferrable._yank(_id, _end) [DssVest], 701

definition

DssVestMintable.tot(_id) [DssVest], 677
 DssVestSuckable.tot(_id) [DssVest], 686
 DssVestTransferrable.tot(_id) [DssVest], 697

read

DssVestMintable.tot(_id) [DssVest], 677
 DssVestMintable.valid(_id) [DssVest], 682
 DssVestSuckable.tot(_id) [DssVest], 686
 DssVestSuckable.valid(_id) [DssVest], 692
 DssVestTransferrable.tot(_id) [DssVest], 697
 DssVestTransferrable.valid(_id) [DssVest], 702

total

write

GemJoin9.exit(usr, wad) **X**, 445
 GemJoin9._join(usr), 444

definition

GemJoin9.total: **uint256**, 443

read

GemJoin9.exit(usr, wad) **X**, 445
 GemJoin9._join(usr), 444

totalBorrows

call

D3MCompoundPlan.getTargetAssets(currentAssets), 317

totalReserves

call

D3MCompoundPlan.getTargetAssets(currentAssets), 317

_totalSupply

call

D3MAavePlan.getTargetAssets(currentAssets), 313

write

Dai.burn(from, **value**) **X**, 590, 616
 Dai.burn(usr, wad) **X**, 225
 Dai.mint(to, **value**) **X a**, 590, 616
 Dai.mint(usr, wad) **X a**, 224
 DAIMock._burn(account, amount) [ERC20], 630
 DAIMock._mint(account, amount) [ERC20], 630

definition

AAVE.totalSupply(), 451
 BAL.totalSupply(), 453
 BAT.totalSupply(), 455
 COMP.totalSupply(), 457
 DAIMock._totalSupply [ERC20] : **uint256**, 625
 DAIMock.totalSupply() [ERC20], 627
 Dai.totalSupply: **uint256**, 223, 587, 613
 DGD.totalSupply(), 459
 GNT.totalSupply(), 461
 GUSD.totalSupply(), 463


```

KNC.totalSupply(), 465
LINK.totalSupply(), 467
LRC.totalSupply(), 469
MANA.totalSupply(), 471
MATIC.totalSupply(), 473
OMG.totalSupply(), 475
PAXG.totalSupply(), 478
PAXUSD.totalSupply(), 481
RENBTC.totalSupply(), 483
REP.totalSupply(), 485
TUSD.totalSupply(), 487
UNI.totalSupply(), 489
USDC.totalSupply(), 491
WBTC.totalSupply(), 495
WETH9_.totalSupply(), 517
WSTETH.totalSupply(), 497
YFI.totalSupply(), 499
ZRX.totalSupply(), 501
read
  Dai.burn(from, value) X, 590, 616
  Dai.burn(usr, wad) X, 225
  Dai.mint(to, value) X a, 590, 616
  Dai.mint(usr, wad) X a, 224
  DAIMock.burn(account, amount) [ERC20], 630
  DAIMock.mint(account, amount) [ERC20], 630
  DAIMock.totalSupply() [ERC20], 627
toUint48
  call
    DssVestMintable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a, 678
    DssVestMintable._yank(_id, _end) [DssVest], 682
    DssVestSuckable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a, 688
    DssVestSuckable._yank(_id, _end) [DssVest], 691
    DssVestTransferrable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a, 698
    DssVestTransferrable._yank(_id, _end) [DssVest], 701
  definition
    DssVestMintable.toUint48(x) [DssVest], 678
    DssVestSuckable.toUint48(x) [DssVest], 688
    DssVestTransferrable.toUint48(x) [DssVest], 698
toUint128
  call
    DssVestMintable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a, 678
    DssVestMintable._vest(_id, _maxAmt) [DssVest], 679
    DssVestMintable._yank(_id, _end) [DssVest], 682
    DssVestSuckable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a, 688
    DssVestSuckable._vest(_id, _maxAmt) [DssVest], 689
    DssVestSuckable._yank(_id, _end) [DssVest], 691
    DssVestTransferrable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a, 698
    DssVestTransferrable._vest(_id, _maxAmt) [DssVest], 699
    DssVestTransferrable._yank(_id, _end) [DssVest], 701
  definition
    DssVestMintable.toUint128(x) [DssVest], 678
    DssVestSuckable.toUint128(x) [DssVest], 688
    DssVestTransferrable.toUint128(x) [DssVest], 698
toUpper
  call
    D3MCoreDeployScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 345
    D3MCoreInitScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 373
    D3MDeployScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 399
    D3MInitScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 419
  definition
    D3MCoreDeployScript._toUpper(str) [StdChains], 345
    D3MCoreInitScript._toUpper(str) [StdChains], 373
    D3MDeployScript._toUpper(str) [StdChains], 399

```

D3MInitScript.toUpper(str) [StdChains], 419

tout

write

DssPsm.file(what, data) X a, 510

definition

DssPsm.tout: uint256, 509

read

DssPsm.buyGem(usr, gemAmt) X, 511

transactionIndex

write

D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341

D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369

D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395

D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415

read

D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341

D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369

D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395

D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415

transactionLogIndex

write

D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341

D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369

D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395

D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415

read

D3MCoreDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 341

D3MCoreInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 369

D3MDeployScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 395

D3MInitScript.rawToConvertedReceiptLogs(rawLogs) [StdCheatsSafe], 415

transfer

call

AuthGemJoin5.exit(usr, amt) X, 504

AuthGemJoin8.exit(usr, amt) X, 506

AuthGemJoin.exit(guy, wad) X, 508

AuthGemJoin.exit(usr, wad) X, 448

BasicRelay.relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s) X, 658

CalleeMaker0tcDai.clipperCall(sender, daiAmt, gemAmt, data) X, 550

CurveLpTokenUniv3Callee.clipperCall(sender, owe, slice, data) X, 552

D3MAavePool.exit(dst, wad) X, 351

D3MAavePool.quit(dst) X a, 351

D3MCompoundPool.collect(claim) X, 380

D3MCompoundPool.exit(dst, wad) X, 379

D3MCompoundPool.quit(dst) X a, 379

D3MCompoundPool.withdraw(wad) X, 379

DAIMock.transfer(to, amount) [ERC20] X, 627

DAIMock.transferFrom(from, to, amount) [ERC20] X, 628

DssProxyActions.transfer(gem, dst, amt) X, 531

DssProxyActions.openLockGNTAndDraw(manager, jug, gntJoin, daiJoin, ilk, amtC, wadD) X, 539

GemBag.exit(usr, wad) X, 431

GemJoin3.exit(usr, amt) X, 430

GemJoin4.exit(usr, wad) X, 434

GemJoin5.exit(usr, amt) X, 436

GemJoin6.exit(usr, wad) X, 438

GemJoin7.exit(usr, amt) X, 440

GemJoin8.exit(usr, amt) X, 442

GemJoin9.exit(usr, wad) X, 445

GemJoin.exit(usr, wad) X, 424

ManagedGemJoin.exit(urn, usr, amt) X a, 450

PSMCallee.clipperCall(sender, owe, slice, data) X, 555

TrustedRelay.relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s) [X](#), 663
 UniswapV2CalleeDai.clipperCall(sender, daiAmt, gemAmt, data) [X](#), 558
 UniswapV2LpTokenCalleeDai.clipperCall(sender, daiAmt, gemAmt, data) [X](#), 560
 UniswapV2LpTokenCalleeDai.swapGemForDai(token, path, to), 560
 UniswapV3Callee.clipperCall(sender, owe, slice, data) [X](#), 567
 WstETHCurveUniv3Callee.clipperCall(sender, owe, slice, data) [X](#), 564

definition

AAVE.transfer(dst, wad) [X](#), 451
 BAL.transfer(dst, wad) [X](#), 453
 BAT.transfer(dst, wad) [X](#), 455
 COMP.transfer(dst, wad) [X](#), 457
 Dai.transfer(dst, wad) [X](#), 224
 Dai.transfer(to, value) [X](#), 588, 614
 DAIMock.transfer(to, amount) [ERC20] [X](#), 627
 DAIMock.transfer(from, to, amount) [ERC20], 629
 DGD.transfer(dst, wad) [X](#), 459
 DssProxyActions.transfer(gem, dst, amt) [X](#), 531
 GNT.transfer(dst, wad) [X](#), 461
 GUSD.transfer(dst, wad) [X](#), 464
 KNC.transfer(dst, wad) [X](#), 465
 LINK.transfer(dst, wad) [X](#), 467
 LRC.transfer(dst, wad) [X](#), 469
 MANA.transfer(dst, wad) [X](#), 471
 MATIC.transfer(dst, wad) [X](#), 473
 OMG.transfer(dst, wad) [X](#), 475
 PAXG.transfer(dst, wad) [X](#), 478
 PAXUSD.transfer(dst, wad) [X](#), 481
 RENBTC.transfer(dst, wad) [X](#), 483
 REP.transfer(dst, wad) [X](#), 485
 TUSD.transfer(dst, wad) [X](#), 488
 UNI.transfer(dst, wad) [X](#), 489
 USDC.transfer(dst, wad) [X](#), 491
 USDT.transfer(_to, _value) [X](#), 493
 WBTC.transfer(dst, wad) [X](#), 495
 WETH9.transfer(dst, wad) [X](#), 517
 WSTETH.transfer(dst, wad) [X](#), 497
 YFI.transfer(dst, wad) [X](#), 499
 ZRX.transfer(dst, wad) [X](#), 501

transferFrom

call

AAVE.transfer(dst, wad) [X](#), 451
 AuthGemJoin5.join(urn, amt, msgSender) [X](#) [a](#), 504
 AuthGemJoin8.join(urn, amt, msgSender) [X](#) [a](#), 506
 AuthGemJoin.join(urn, wad, msgSender) [X](#) [a](#), 508
 AuthGemJoin.join(usr, wad) [X](#) [a](#), 448
 BAL.transfer(dst, wad) [X](#), 453
 BAT.transfer(dst, wad) [X](#), 455
 COMP.transfer(dst, wad) [X](#), 457
 Dai.push(usr, wad) [X](#), 225
 Dai.transfer(dst, wad) [X](#), 224
 Dai.move(src, dst, wad) [X](#), 225
 Dai.pull(usr, wad) [X](#), 225
 DGD.transfer(dst, wad) [X](#), 459
 DssProxyActions.daiJoin_join(apt, urn, wad) [Common] [X](#), 529
 DssProxyActionsDsr.daiJoin_join(apt, urn, wad) [Common] [X](#), 547
 DssProxyActionsEnd.daiJoin_join(apt, urn, wad) [Common] [X](#), 543
 DssProxyActions.gemJoin_join(apt, urn, amt, transferFrom) [X](#), 531
 DssPsm.buyGem(usr, gemAmt) [X](#), 511
 DssVestTransferrable.pay(_guy, _amt), 703
 ESM.join(wad) [X](#), 305
 GemJoin3.join(usr, amt) [X](#), 430
 GemJoin5.join(usr, amt) [X](#), 436

GemJoin6.join(usr, wad) [X](#), 438
 GemJoin7.join(usr, amt) [X](#), 440
 GemJoin8.join(usr, amt) [X](#), 442
 GemJoin9.join(usr, wad) [X](#), 444
 GemJoin.join(usr, wad) [X](#), 424
 GUSD.transfer(dst, wad) [X](#), 464
 KNC.transfer(dst, wad) [X](#), 465
 L1DAIBridge.cancelDeposit(amount, l2Recipient, l1Recipient, nonce) [X](#), 620
 L1DAIBridge.deposit(amount, l2Recipient) [X](#), 619
 L1DAIBridge.withdraw(amount, l1Recipient) [X](#), 619
 L1DaiGateway.finalizeInboundTransfer(l1Token, from, to, amount, data) [X](#), 572
 L1DaiGateway.outboundTransfer(l1Token, to, amount, maxGas, gasPriceBid, data) [X](#), 571
 L1DAITeleportGateway.finalizeFlush(targetDomain, daiToFlush) [X](#), 623
 L1DAITokenBridge.finalizeERC20Withdrawal(_l1Token, _l2Token, _from, _to, _amount, _data) [X](#), 595
 L1DAITokenBridge._initiateERC20Deposit(_from, _to, _amount, _l2Gas, _data), 594
 LINK.transfer(dst, wad) [X](#), 467
 LRC.transfer(dst, wad) [X](#), 469
 MANA.transfer(dst, wad) [X](#), 471
 ManagedGemJoin.join(usr, amt) [X](#) [a](#), 450
 MATIC.transfer(dst, wad) [X](#), 473
 OMG.transfer(dst, wad) [X](#), 475
 PAXG.transfer(dst, wad) [X](#), 478
 PAXUSD.transfer(dst, wad) [X](#), 481
 RENBTC.transfer(dst, wad) [X](#), 483
 REP.transfer(dst, wad) [X](#), 485
 TeleportRouter.settle(targetDomain, batchedDaiToFlush) [X](#), 655
 TUSD.transfer(dst, wad) [X](#), 488
 UNI.transfer(dst, wad) [X](#), 489
 USDC.transfer(dst, wad) [X](#), 491
 WBTC.transfer(dst, wad) [X](#), 495
 WETH9_.transfer(dst, wad) [X](#), 517
 WSTETH.transfer(dst, wad) [X](#), 497
 YFI.transfer(dst, wad) [X](#), 499
 ZRX.transfer(dst, wad) [X](#), 501

definition

AAVE.transferFrom(src, dst, wad) [X](#), 452
 BAL.transferFrom(src, dst, wad) [X](#), 454
 BAT.transferFrom(src, dst, wad) [X](#), 456
 COMP.transferFrom(src, dst, wad) [X](#), 458
 DAIMock.transferFrom(from, to, amount) [ERC20] [X](#), 628
 Dai.transferFrom(from, to, value) [X](#), 589, 615
 Dai.transferFrom(src, dst, wad) [X](#), 224
 DGD.transferFrom(src, dst, wad) [X](#), 460
 GUSD.transferFrom(src, dst, wad) [X](#), 464
 KNC.transferFrom(src, dst, wad) [X](#), 466
 LINK.transferFrom(src, dst, wad) [X](#), 468
 LRC.transferFrom(src, dst, wad) [X](#), 470
 MANA.transferFrom(src, dst, wad) [X](#), 472
 MATIC.transferFrom(src, dst, wad) [X](#), 474
 OMG.transferFrom(src, dst, wad) [X](#), 476
 PAXG.transferFrom(src, dst, wad) [X](#), 478
 PAXUSD.transferFrom(src, dst, wad) [X](#), 482
 RENBTC.transferFrom(src, dst, wad) [X](#), 484
 REP.transferFrom(src, dst, wad) [X](#), 486
 TUSD.transferFrom(src, dst, wad) [X](#), 488
 UNI.transferFrom(src, dst, wad) [X](#), 490
 USDC.transferFrom(src, dst, wad) [X](#), 492
 USDT.transferFrom(_from, _to, _value) [X](#), 494
 WBTC.transferFrom(src, dst, wad) [X](#), 496
 WETH9_.transferFrom(src, dst, wad) [X](#), 518
 WSTETH.transferFrom(src, dst, wad) [X](#), 498

```

        YFI.transferFrom(src, dst, wad) X, 500
        ZRX.transferFrom(src, dst, wad) X, 502
tripBreaker
    definition
        ClipperMom.tripBreaker(clip) X, 259
TrustedRelay
    definition
        contract TrustedRelay, 661
ttl
    write
        Flapper.file(what, data) X a, 263
        Flopper.file(what, data) X a, 266
        TeleportConstantFee.constructor(_fee, _ttl) X, 639
        TeleportLinearFee.constructor(_fee, _ttl) X, 647
    definition
        Flapper.ttl: uint48, 261
        Flopper.ttl: uint48, 265
        TeleportConstantFee.ttl: uint256, 639
        TeleportLinearFee.ttl: uint256, 647
    read
        DssAutoLine.exec(_ilk) X, 707
        Flapper.tend(id, lot, bid) X, 263
        Flopper.dend(id, lot, bid) X, 267
        TeleportConstantFee.getFee(guid, , , , amtToTake), 639
        TeleportLinearFee.getFee(guid, , , , amtToTake), 647
TUSD
    definition
        contract TUSD, 487
TWENTY_YEARS
    definition
        DssVestMintable.TWENTY_YEARS [DssVest] : uint256, 675
        DssVestSuckable.TWENTY_YEARS [DssVest] : uint256, 685
        DssVestTransferrable.TWENTY_YEARS [DssVest] : uint256, 695
    read
        DssVestMintable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a, 678
        DssVestSuckable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a, 688
        DssVestTransferrable.create(_usr, _tot, _bgn, _tau, _eta, _mgr) [DssVest] X a, 698
Tx1559
    definition
        struct StdCheatsSafe.Tx1559, 332, 360, 386, 406
Tx1559Detail
    definition
        struct StdCheatsSafe.Tx1559Detail, 332, 360, 386, 406
TxDetailLegacy
    definition
        struct StdCheatsSafe.TxDetailLegacy, 332, 360, 386, 406
TxLegacy
    definition
        struct StdCheatsSafe.TxLegacy, 332, 360, 386, 406
TxReturn
    definition
        struct StdCheatsSafe.TxReturn, 334, 362, 388, 408

```

U

UINT256_MAX

definition

D3MCoreDeployScript.UINT256_MAX [CommonBase] : uint256, 329
 D3MCoreDeployScript.UINT256_MAX [StdUtils] : uint256, 329
 D3MCoreInitScript.UINT256_MAX [CommonBase] : uint256, 357
 D3MCoreInitScript.UINT256_MAX [StdUtils] : uint256, 357
 D3MDeployScript.UINT256_MAX [CommonBase] : uint256, 383
 D3MDeployScript.UINT256_MAX [StdUtils] : uint256, 383
 D3MInitScript.UINT256_MAX [CommonBase] : uint256, 403
 D3MInitScript.UINT256_MAX [StdUtils] : uint256, 403

read

D3MCoreDeployScript._bound(x, min, max) [StdUtils], 335
 D3MCoreInitScript._bound(x, min, max) [StdUtils], 363
 D3MDeployScript._bound(x, min, max) [StdUtils], 389
 D3MInitScript._bound(x, min, max) [StdUtils], 409

UintSet

definition

struct EnumerableSet.UintSet, 756

uncull

definition

D3MHub.uncull(ilk) X, 326

undelegateVat

definition

DssExecLib.undelegateVat(_usr) X, 734

underlying

call

D3MCompoundPool.constructor(ilk_, hub_, cDai_) X, 377

UNI

definition

contract UNI, 489

uniRouter02

write

UniswapV2CalleeDai.setUp(uniRouter02_, daiJoin_) [UniswapV2Callee], 557
 UniswapV2LpTokenCalleeDai.setUp(uniRouter02_, daiJoin_) [UniswapV2Callee(2)], 559

definition

UniswapV2CalleeDai.uniRouter02 [UniswapV2Callee] : UniswapV2Router02Like, 557
 UniswapV2LpTokenCalleeDai.uniRouter02 [UniswapV2Callee(2)] :
 UniswapV2Router02Like, 559

read

UniswapV2CalleeDai.clipperCall(sender, daiAmt, gemAmt, data) X, 558
 UniswapV2LpTokenCalleeDai.clipperCall(sender, daiAmt, gemAmt, data) X, 560
 UniswapV2LpTokenCalleeDai.swapGemForDai(token, path, to), 560

UniswapV2CalleeDai

definition

contract UniswapV2CalleeDai, 557

UniswapV2LpTokenCalleeDai

definition

contract UniswapV2LpTokenCalleeDai, 559

UniswapV3Callee

definition

contract UniswapV3Callee, 567

uniV3Router

write

CurveLpTokenUniv3Callee.constructor(uniV3Router_, daiJoin_, weth_) X, 551
 UniswapV3Callee.constructor(uniV3Router_, daiJoin_) X, 567
 WstETHCurveUniv3Callee.constructor(curvePool_, uniV3Router_, daiJoin_, weth_) X,
 563

definition

CurveLpTokenUniv3Callee.uniV3Router: UniV3RouterLike, 551
 UniswapV3Callee.uniV3Router: UniV3RouterLike, 567
 WstETHCurveUniv3Callee.uniV3Router: UniV3RouterLike, 563

read

```

CurveLpTokenUniv3Callee.clipperCall(sender, owe, slice, data) X, 552
UniswapV3Callee.clipperCall(sender, owe, slice, data) X, 567
WstETHCurveUniv3Callee.clipperCall(sender, owe, slice, data) X, 564
unpaid
  call
    DssVestMintable.unpaid(_id) [DssVest], 680
    DssVestMintable._vest(_id, _maxAmt) [DssVest], 679
    DssVestMintable._yank(_id, _end) [DssVest], 682
    DssVestSuckable.unpaid(_id) [DssVest], 690
    DssVestSuckable._vest(_id, _maxAmt) [DssVest], 689
    DssVestSuckable._yank(_id, _end) [DssVest], 691
    DssVestTransferrable.unpaid(_id) [DssVest], 700
    DssVestTransferrable._vest(_id, _maxAmt) [DssVest], 699
    DssVestTransferrable._yank(_id, _end) [DssVest], 701
  definition
    DssVestMintable.unpaid(_id) [DssVest], 680
    DssVestMintable.unpaid(_time, _bgn, _clf, _fin, _tot, _rx) [DssVest], 680
    DssVestSuckable.unpaid(_id) [DssVest], 690
    DssVestSuckable.unpaid(_time, _bgn, _clf, _fin, _tot, _rx) [DssVest], 690
    DssVestTransferrable.unpaid(_id) [DssVest], 700
    DssVestTransferrable.unpaid(_time, _bgn, _clf, _fin, _tot, _rx) [DssVest], 700
unrestrict
  definition
    DssVestMintable.unrestrict(_id) [DssVest] X, 681
    DssVestSuckable.unrestrict(_id) [DssVest] X, 691
    DssVestTransferrable.unrestrict(_id) [DssVest] X, 701
unwrap
  call
    WstETHCurveUniv3Callee.clipperCall(sender, owe, slice, data) X, 564
upchost
  definition
    Clipper.upchost() X, 248
update
  definition
    IlkRegistry.update(ilk) X, 673
updateCollateralPrice
  call
    DssExecLib.addNewCollateral(co) X, 750
    DssExecLib.setRWAILkDebtCeiling(_ilk, _ceiling, _price) X, 742
  definition
    DssExecLib.updateCollateralPrice(_ilk) X, 736
upgradedAddress
  call
    GemJoin7.constructor(vat_, ilk_, gem_) X, 439
    GemJoin7.exit(usr, amt) X, 440
    GemJoin7.join(usr, amt) X, 440
  write
    USDT.deprecate(_upgradedAddress) X, 494
  definition
    USDT.upgradedAddress: address, 493
Urn
  definition
    struct Vat.Urn, 229
urnAllow
  call
    DssProxyActions.urnAllow(manager, usr, ok) X, 533
  definition
    DssCdpManager.urnAllow(usr, ok) X, 522
    DssProxyActions.urnAllow(manager, usr, ok) X, 533
urnAllowed
  call
    DssCdpManager.enter(src, cdp) X, 525
    DssCdpManager.quit(cdp, dst) X, 524

```


definition

DssCdpManager.modifier urnAllowed(urn), 521

urnCan

write

DssCdpManager.urnAllow(usr, ok) [X](#), 522

definition

DssCdpManager.urnCan: [mapping](#)(address => [mapping](#)(address => uint256)), 521

read

DssCdpManager.modifier urnAllowed(urn), 521

UrnHandler

definition

[contract](#) UrnHandler, 519

urns

call

D3MHub.cull(ilk) [X](#), 326

D3MHub._exec(ilk, _pool, Art, lineWad), 322

Dog.bark(ilk, urn, kpr) [X](#), 253

DssCdpManager.enter(src, cdp) [X](#), 525

DssCdpManager.quit(cdp, dst) [X](#), 524

DssCdpManager.shift(cdpSrc, cdpDst) [X](#), 525

DssProxyActions.draw(manager, jug, daiJoin, cdp, wad) [X](#), 536

DssProxyActionsEnd._free(manager, end, cdp), 543

DssProxyActions._getWipeAllWad(vat, usr, urn, ilk), 530

DssProxyActions._getWipeDart(vat, dai, urn, ilk), 530

DssProxyActions.lockETH(manager, ethJoin, cdp) [X](#), 534

DssProxyActions.lockETHAndDraw(manager, jug, ethJoin, daiJoin, cdp, wadD) [X](#), 538

DssProxyActions.lockGem(manager, gemJoin, cdp, amt, transferFrom) [X](#), 535

DssProxyActions.lockGemAndDraw(manager, jug, gemJoin, daiJoin, cdp, amtC, wadD, transferFrom) [X](#), 539

DssProxyActions.wipe(manager, daiJoin, cdp, wad) [X](#), 537

DssProxyActions.wipeAll(manager, daiJoin, cdp) [X](#), 537

DssProxyActions.wipeAllAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC) [X](#), 540

DssProxyActions.wipeAllAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC) [X](#), 541

DssProxyActions.wipeAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC, wadD) [X](#), 540

DssProxyActions.wipeAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC, wadD) [X](#), 541

End.free(ilk) [X](#), 300

End.skim(ilk, urn) [X](#), 300

TeleportJoin._mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643

TeleportJoin.settle(sourceDomain, batchedDaiToFlush) [X](#), 645

write

DssCdpManager.open(ilk, usr) [X](#), 522

Vat.fork(ilk, src, dst, dink, dart) [X](#), 233

Vat.frob(i, u, v, w, dink, dart) [X](#), 232

Vat.grab(i, u, v, w, dink, dart) [X](#) [a](#), 233

definition

DssCdpManager.urns: [mapping](#)(uint256 => [address](#)), 521

Vat.urns: [mapping](#)(bytes32 => [mapping](#)([address](#) => Vat.Urn)), 229

read

DssCdpManager.enter(src, cdp) [X](#), 525

DssCdpManager.flux(cdp, dst, wad) [X](#), 524

DssCdpManager.flux(ilk, cdp, dst, wad) [X](#), 524

DssCdpManager.frob(cdp, dink, dart) [X](#), 524

DssCdpManager.move(cdp, dst, rad) [X](#), 524

DssCdpManager.quit(cdp, dst) [X](#), 524

DssCdpManager.shift(cdpSrc, cdpDst) [X](#), 525

GetCdps.getCdpsAsc(manager, guy), 527

GetCdps.getCdpsDesc(manager, guy), 527

Vat.fork(ilk, src, dst, dink, dart) [X](#), 233

Vat.frob(i, u, v, w, dink, dart) [X](#), 232

Vat.grab(i, u, v, w, dink, dart) [X](#) [a](#), 233

USDC

definition

[contract](#) USDC, 491

USDT

definition

contract USDT, 493

usr

write

Clipper.kick(tab, lot, usr, kpr) X a, 244
DssVestMintable.move(_id, _dst) [DssVest] X, 682
DssVestSuckable.move(_id, _dst) [DssVest] X, 692
DssVestTransferrable.move(_id, _dst) [DssVest] X, 702

definition

DssVestMintable.usr(_id) [DssVest], 676
DssVestSuckable.usr(_id) [DssVest], 686
DssVestTransferrable.usr(_id) [DssVest], 696

read

Clipper.getStatus(id), 248
Clipper.redo(id, kpr) X, 245
Clipper.take(id, amt, max, who, data) X, 246
Clipper.yank(id) X a, 249
DssVestMintable.move(_id, _dst) [DssVest] X, 682
DssVestMintable.restrict(_id) [DssVest] X, 681
DssVestMintable.unrestrict(_id) [DssVest] X, 681
DssVestMintable.usr(_id) [DssVest], 676
DssVestSuckable.move(_id, _dst) [DssVest] X, 692
DssVestSuckable.restrict(_id) [DssVest] X, 690
DssVestSuckable.unrestrict(_id) [DssVest] X, 691
DssVestSuckable.usr(_id) [DssVest], 686
DssVestTransferrable.move(_id, _dst) [DssVest] X, 702
DssVestTransferrable.restrict(_id) [DssVest] X, 701
DssVestTransferrable.unrestrict(_id) [DssVest] X, 701
DssVestTransferrable.usr(_id) [DssVest], 696

V

val

- write
 - Median.poke(val_, age_, v, r, s) [X](#), 283
- definition
 - Median.val: [uint128](#), 281
 - Value.val: [bytes32](#), 289
- read
 - Median.peak(), 282
 - Median.poke(val_, age_, v, r, s) [X](#), 283
 - Median.read(), 282
 - Value.peak(), 289
- valid
 - definition
 - DssVestMintable.valid(_id) [DssVest], 682
 - DssVestSuckable.valid(_id) [DssVest], 692
 - DssVestTransferrable.valid(_id) [DssVest], 702
- Value
 - definition
 - [contract](#) Value, 289
- _values
 - call
 - EnumerableSet.values(set), 759–761
 - definition
 - EnumerableSet._values(set), 757, 759–761
- variableDebt
 - call
 - D3MInit.initAave(dss, d3m, cfg, aaveCfg), 726
 - D3MInitScript.run() [X](#), 420
 - write
 - D3MAavePlan.[constructor](#)(dai_, pool_) [X](#), 311
 - D3MAavePool.[constructor](#)(ilk_, hub_, dai_, pool_) [X](#), 349
 - definition
 - D3MAavePlan.variableDebt: [TokenLike](#), 311
 - D3MAavePool.variableDebt: [ATokenLike](#), 349
 - read
 - D3MAavePlan.active(), 314
 - D3MAavePlan.getTargetAssets(currentAssets), 313
- variableRateSlope1
 - call
 - D3MAavePlan._calculateTargetSupply(targetInterestRate, totalDebt), 313
- variableRateSlope2
 - call
 - D3MAavePlan._calculateTargetSupply(targetInterestRate, totalDebt), 313
- Vat
 - definition
 - [contract](#) Vat, 229
- vat
 - call
 - D3MAavePool.[constructor](#)(ilk_, hub_, dai_, pool_) [X](#), 349
 - D3MCompoundPool.[constructor](#)(ilk_, hub_, cDai_) [X](#), 377
 - D3MHub.[constructor](#)(daiJoin_) [X](#), 320
 - D3MInit._init(dss, d3m, cfg, gem), 725
 - D3MInit.initAave(dss, d3m, cfg, aaveCfg), 726
 - D3MInit.initCompound(dss, d3m, cfg, compoundCfg), 727
 - D3MInit.initCore(dss, d3mCore), 725
 - DssExecLib.addCollateralBase(ilk_, gem, _join, _clip, _calc, _pip) [X](#), 749
 - DssExecLib.decreaseGlobalDebtCeiling(_amount) [X](#), 737
 - DssExecLib.decreaseIlkDebtCeiling(_ilk, _amount, _global) [X](#), 742
 - DssExecLib.delegateVat(_usr) [X](#), 734
 - DssExecLib.increaseGlobalDebtCeiling(_amount) [X](#), 737
 - DssExecLib.increaseIlkDebtCeiling(_ilk, _amount, _global) [X](#), 741
 - DssExecLib.sendPaymentFromSurplusBuffer(_target, _amount) [X](#), 751

DssExecLib.setGlobalDebtCeiling(_amount) [X](#), 737
 DssExecLib.setIlkDebtCeiling(_ilk, _amount) [X](#), 741
 DssExecLib.setIlkMinVaultAmount(_ilk, _amount) [X](#), 743
 DssExecLib.undelegateVat(_usr) [X](#), 734
 DssProxyActions.draw(manager, jug, daiJoin, cdp, wad) [X](#), 536
 DssProxyActionsDsr.exit(daiJoin, pot, wad) [X](#), 548
 DssProxyActionsDsr.exitAll(daiJoin, pot) [X](#), 548
 DssProxyActionsDsr.join(daiJoin, pot, wad) [X](#), 547
 DssProxyActionsEnd._free(manager, end, cdp), 543
 DssProxyActionsEnd.pack(daiJoin, end, wad) [X](#), 544
 DssProxyActions.lockETH(manager, ethJoin, cdp) [X](#), 534
 DssProxyActions.lockETHAndDraw(manager, jug, ethJoin, daiJoin, cdp, wadD) [X](#), 538
 DssProxyActions.lockGem(manager, gemJoin, cdp, amt, transferFrom) [X](#), 535
 DssProxyActions.lockGemAndDraw(manager, jug, gemJoin, daiJoin, cdp, amtC, wadD, transferFrom) [X](#), 539
 DssProxyActions.wipe(manager, daiJoin, cdp, wad) [X](#), 537
 DssProxyActions.wipeAll(manager, daiJoin, cdp) [X](#), 537
 DssProxyActions.wipeAllAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC) [X](#), 540
 DssProxyActions.wipeAllAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC) [X](#), 541
 DssProxyActions.wipeAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC, wadD) [X](#), 540
 DssProxyActions.wipeAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC, wadD) [X](#), 541
 DssPsm.constructor(gemJoin_, daiJoin_, vow_) [X](#), 509
 ESM.fire() [X](#), 304
 IlkRegistry.add(adapter) [X](#), 668
 IlkRegistry.constructor(vat_, dog_, cat_, spot_) [X](#), 668
 IlkRegistry.update(ilk) [X](#), 673

write

AuthGemJoin5.constructor(vat_, ilk_, gem_) [X](#), 503
 AuthGemJoin8.constructor(vat_, ilk_, gem_) [X](#), 505
 AuthGemJoin.constructor(vat_, ilk_, gem_) [X](#), 447, 507
 Clipper.constructor(vat_, spotter_, dog_, ilk_) [X](#), 243
 D3MAavePool.constructor(ilk_, hub_, dai_, pool_) [X](#), 349
 D3MCompoundPool.constructor(ilk_, hub_, cDai_) [X](#), 377
 D3MHub.constructor(daiJoin_) [X](#), 320
 D3MOracle.constructor(vat_, ilk_) [X](#), 353
 DaiJoin.constructor(vat_, dai_) [X](#), 425
 Dog.constructor(vat_) [X](#), 252
 DssAutoLine.constructor(vat_) [X](#), 705
 DssCdpManager.constructor(vat_) [X](#), 522
 DssPsm.constructor(gemJoin_, daiJoin_, vow_) [X](#), 509
 DssVestSuckable.constructor(_chainlog) [X](#), 692
 End.file(what, data) [X](#) a, 298
 Flapper.constructor(vat_, gem_) [X](#), 262
 Flopper.constructor(vat_, gem_) [X](#), 266
 GemJoin2.constructor(vat_, ilk_, gem_) [X](#), 427
 GemJoin3.constructor(vat_, ilk_, gem_, decimals) [X](#), 429
 GemJoin4.constructor(vat_, ilk_, gem_) [X](#), 433
 GemJoin5.constructor(vat_, ilk_, gem_) [X](#), 435
 GemJoin6.constructor(vat_, ilk_, gem_) [X](#), 437
 GemJoin7.constructor(vat_, ilk_, gem_) [X](#), 439
 GemJoin8.constructor(vat_, ilk_, gem_) [X](#), 441
 GemJoin9.constructor(vat_, ilk_, gem_) [X](#), 443
 GemJoin.constructor(vat_, ilk_, gem_) [X](#), 424
 IlkRegistry.constructor(vat_, dog_, cat_, spot_) [X](#), 668
 Jug.constructor(vat_) [X](#), 273
 ManagedGemJoin.constructor(vat_, ilk_, gem_) [X](#), 449
 Pot.constructor(vat_) [X](#), 277
 Spotter.constructor(vat_) [X](#), 227
 TeleportJoin.constructor(vat_, daiJoin_, ilk_, domain_) [X](#), 641
 Vow.constructor(vat_, flapper_, flopper_) [X](#), 269

definition

AuthGemJoin5.vat: VatLike, 503
 AuthGemJoin8.vat: VatLike, 505

AuthGemJoin.vat: VatLike, 447, 507
 Clipper.vat: VatLike, 241
 D3MAavePool.vat: VatLike, 349
 D3MCompoundPool.vat: VatLike, 377
 D3MHub.vat: VatLike, 319
 D3MOracle.vat: [address](#), 353
 DaiJoin.vat: VatLike, 425
 Dog.vat: VatLike, 251
 DssAutoLine.vat: VatLike, 705
 DssCdpManager.vat: [address](#), 521
 DssExecLib.vat(), 730
 DssPsm.vat: VatAbstract, 509
 DssVestSuckable.vat: VatLike, 685
 End.vat: VatLike, 295
 Flapper.vat: VatLike, 261
 Flopper.vat: VatLike, 265
 GemJoin2.vat: VatLike, 427
 GemJoin3.vat: VatLike, 429
 GemJoin4.vat: VatLike, 433
 GemJoin5.vat: VatLike, 435
 GemJoin6.vat: VatLike, 437
 GemJoin7.vat: VatLike, 439
 GemJoin8.vat: VatLike, 441
 GemJoin9.vat: VatLike, 443
 GemJoin.vat: VatLike, 423
 IlkRegistry.vat: VatLike, 667
 Jug.vat: VatLike, 273
 ManagedGemJoin.vat: VatLike, 449
 Pot.vat: VatLike, 277
 Spotter.vat: VatLike, 227
 TeleportJoin.vat: VatLike, 641
 Vow.vat: VatLike, 269

read

AuthGemJoin5.exit(usr, amt) [X](#), 504
 AuthGemJoin5.join(urn, amt, msgSender) [X](#) [a](#), 504
 AuthGemJoin8.exit(usr, amt) [X](#), 506
 AuthGemJoin8.join(urn, amt, msgSender) [X](#) [a](#), 506
 AuthGemJoin.exit(guy, wad) [X](#), 508
 AuthGemJoin.exit(usr, wad) [X](#), 448
 AuthGemJoin.join(urn, wad, msgSender) [X](#) [a](#), 508
 AuthGemJoin.join(usr, wad) [X](#) [a](#), 448
 Clipper.kick(tab, lot, usr, kpr) [X](#) [a](#), 244
 Clipper.redo(id, kpr) [X](#), 245
 Clipper.take(id, amt, max, who, [data](#)) [X](#), 246
 Clipper.upchost() [X](#), 248
 Clipper.yank(id) [X](#) [a](#), 249
 D3MAavePool.[constructor](#)(ilk_, hub_, dai_, pool_) [X](#), 349
 D3MAavePool.file(what, [data](#)) [X](#) [a](#), 350
 D3MAavePool.quit(dst) [X](#) [a](#), 351
 D3MCompoundPool.[constructor](#)(ilk_, hub_, cDai_) [X](#), 377
 D3MCompoundPool.file(what, [data](#)) [X](#) [a](#), 378
 D3MCompoundPool.quit(dst) [X](#) [a](#), 379
 D3MHub.[constructor](#)(daiJoin_) [X](#), 320
 D3MHub.cage(ilk) [X](#) [a](#), 325
 D3MHub.cull(ilk) [X](#), 326
 D3MHub.exec(ilk) [X](#), 324
 D3MHub._exec(ilk, _pool, Art, lineWad), 322
 D3MHub.exit(ilk, usr, wad) [X](#), 325
 D3MHub.file(ilk, what, [data](#)) [X](#) [a](#), 321
 D3MHub.file(what, [data](#)) [X](#) [a](#), 321
 D3MHub.uncull(ilk) [X](#), 326
 D3MHub._wipe(ilk, _pool, urn), 322
 D3MOracle.file(what, [data](#)) [X](#) [a](#), 354

D3MOracle.peek(), 354
DaiJoin.exit(usr, wad) [X](#), 426
DaiJoin.join(usr, wad) [X](#), 426
Dog.bark(ilk, urn, kpr) [X](#), 253
DssAutoLine.exec(_ilk) [X](#), 707
DssCdpManager.enter(src, cdp) [X](#), 525
DssCdpManager.flux(cdp, dst, wad) [X](#), 524
DssCdpManager.flux(ilk, cdp, dst, wad) [X](#), 524
DssCdpManager.frob(cdp, dink, dart) [X](#), 524
DssCdpManager.move(cdp, dst, rad) [X](#), 524
DssCdpManager.open(ilk, usr) [X](#), 522
DssCdpManager.quit(cdp, dst) [X](#), 524
DssCdpManager.shift(cdpSrc, cdpDst) [X](#), 525
DssPsm.[constructor](#)(gemJoin_, daiJoin_, vow_) [X](#), 509
DssPsm.buyGem(usr, gemAmt) [X](#), 511
DssPsm.hope(usr) [X](#) [a](#), 510
DssPsm.nope(usr) [X](#) [a](#), 510
DssPsm.sellGem(usr, gemAmt) [X](#), 510
DssVestSuckable.[constructor](#)(_chainlog) [X](#), 692
DssVestSuckable.pay(_guy, _amt), 693
End.cage() [X](#) [a](#), 299
End.cage(ilk) [X](#), 299
End.cash(ilk, wad) [X](#), 301
End.flow(ilk) [X](#), 300
End.free(ilk) [X](#), 300
End.pack(wad) [X](#), 301
End.skim(ilk, urn) [X](#), 300
End.skip(ilk, id) [X](#), 299
End.snip(ilk, id) [X](#), 299
End.thaw() [X](#), 300
Flapper.cage(rad) [X](#) [a](#), 264
Flapper.deal(id) [X](#), 264
Flapper.kick(lot, bid) [X](#) [a](#), 263
Flopper.dent(id, lot, bid) [X](#), 267
Flopper.yank(id) [X](#), 267
GemJoin2.exit(usr, wad) [X](#), 428
GemJoin2.join(usr, wad) [X](#), 428
GemJoin3.exit(usr, amt) [X](#), 430
GemJoin3.join(usr, amt) [X](#), 430
GemJoin4.exit(usr, wad) [X](#), 434
GemJoin4.join(usr, wad) [X](#), 434
GemJoin5.exit(usr, amt) [X](#), 436
GemJoin5.join(usr, amt) [X](#), 436
GemJoin6.exit(usr, wad) [X](#), 438
GemJoin6.join(usr, wad) [X](#), 438
GemJoin7.exit(usr, amt) [X](#), 440
GemJoin7.join(usr, amt) [X](#), 440
GemJoin8.exit(usr, amt) [X](#), 442
GemJoin8.join(usr, amt) [X](#), 442
GemJoin9.exit(usr, wad) [X](#), 445
GemJoin9._join(usr), 444
GemJoin.exit(usr, wad) [X](#), 424
GemJoin.join(usr, wad) [X](#), 424
IlkRegistry.add(adapter) [X](#), 668
IlkRegistry.[constructor](#)(vat_, dog_, cat_, spot_) [X](#), 668
IlkRegistry.update(ilk) [X](#), 673
Jug.drip(ilk) [X](#), 275
ManagedGemJoin.exit(urn, usr, amt) [X](#) [a](#), 450
ManagedGemJoin.join(usr, amt) [X](#) [a](#), 450
Pot.drip() [X](#), 279
Pot.exit(wad) [X](#), 279
Pot.join(wad) [X](#), 279
Spotter.poke(ilk) [X](#), 228

TeleportJoin.constructor(vat_, daiJoin_, ilk_, domain_) X, 641
 TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643
 TeleportJoin.settle(sourceDomain, batchedDaiToFlush) X, 645
 Vow.constructor(vat_, flapper_, flopper_) X, 269
 Vow.cage() X a, 271
 Vow.file(what, data) X a, 270
 Vow.flap() X, 271
 Vow.flop() X, 271
 Vow.heal(rad) X, 270
 Vow.kiss(rad) X, 271
 _verifyCallResult
 call
 Address.functionCallWithValue(target, data, value, errorMessage), 765
 Address.functionDelegateCall(target, data, errorMessage), 766
 Address.functionStaticCall(target, data, errorMessage), 765
 definition
 Address._verifyCallResult(success, returndata, errorMessage), 766
 version
 write
 ChainLog.setVersion(_version) X a, 710
 definition
 ChainLog.version: string, 709
 Dai.version: string, 223, 587, 613
 read
 Dai.constructor(chainId_) X, 224
 Dai._calculateDomainSeparator(chainId), 588, 614
 _vest
 call
 DssVestMintable.vest(_id) [DssVest] X, 679
 DssVestMintable.vest(_id, _maxAmt) [DssVest] X, 679
 DssVestSuckable.vest(_id) [DssVest] X, 689
 DssVestSuckable.vest(_id, _maxAmt) [DssVest] X, 689
 DssVestTransferrable.vest(_id) [DssVest] X, 699
 DssVestTransferrable.vest(_id, _maxAmt) [DssVest] X, 699
 definition
 DssVestMintable.vest(_id) [DssVest] X, 679
 DssVestMintable._vest(_id, _maxAmt) [DssVest], 679
 DssVestMintable.vest(_id, _maxAmt) [DssVest] X, 679
 DssVestSuckable.vest(_id) [DssVest] X, 689
 DssVestSuckable._vest(_id, _maxAmt) [DssVest], 689
 DssVestSuckable.vest(_id, _maxAmt) [DssVest] X, 689
 DssVestTransferrable.vest(_id) [DssVest] X, 699
 DssVestTransferrable._vest(_id, _maxAmt) [DssVest], 699
 DssVestTransferrable.vest(_id, _maxAmt) [DssVest] X, 699
 vice
 write
 Vat.grab(i, u, v, w, dink, dart) X a, 233
 Vat.heal(rad) X, 233
 Vat.suck(u, v, rad) X a, 234
 definition
 Vat.vice: uint256, 229
 read
 Vat.grab(i, u, v, w, dink, dart) X a, 233
 Vat.heal(rad) X, 233
 Vat.suck(u, v, rad) X a, 234
 vm
 call
 GodMode.setBalance(token, who, amount), 768
 GodMode.setWard(base, target, val), 767
 definition
 D3MCoreDeployScript.vm [CommonBase] : Vm, 329
 D3MCoreDeployScript.vm [StdChains] : VmSafe, 330
 D3MCoreDeployScript.vm [StdCheatsSafe] : Vm, 330

D3MCoreDeployScript.vm [StdUtils] : VmSafe, 329
D3MCoreInitScript.vm [CommonBase] : Vm, 357
D3MCoreInitScript.vm [StdChains] : VmSafe, 358
D3MCoreInitScript.vm [StdCheatsSafe] : Vm, 358
D3MCoreInitScript.vm [StdUtils] : VmSafe, 357
D3MDeployScript.vm [CommonBase] : Vm, 383
D3MDeployScript.vm [StdChains] : VmSafe, 384
D3MDeployScript.vm [StdCheatsSafe] : Vm, 384
D3MDeployScript.vm [StdUtils] : VmSafe, 383
D3MInitScript.vm [CommonBase] : Vm, 403
D3MInitScript.vm [StdChains] : VmSafe, 404
D3MInitScript.vm [StdCheatsSafe] : Vm, 404
D3MInitScript.vm [StdUtils] : VmSafe, 403
GodMode.vm(), 767
ScriptTools.vm: VmSafe, 713

read

D3MCoreDeployScript.assumeNoPrecompiles(addr, chainId) [StdCheatsSafe], 338
D3MCoreDeployScript.modifier noGasMetering() [StdCheatsSafe], 335
D3MCoreDeployScript.bound(x, min, max) [StdUtils], 336
D3MCoreDeployScript.deployCode(what) [StdCheatsSafe], 342
D3MCoreDeployScript.deployCode(what, args) [StdCheatsSafe], 341
D3MCoreDeployScript.deployCode(what, args, val) [StdCheatsSafe], 342
D3MCoreDeployScript.deployCode(what, val) [StdCheatsSafe], 342
D3MCoreDeployScript.deriveRememberKey(mnemonic, index) [StdCheatsSafe], 343
D3MCoreDeployScript.getChain(chainId) [StdChains], 343
D3MCoreDeployScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 345
D3MCoreDeployScript.isFork() [StdCheatsSafe], 343
D3MCoreDeployScript.makeAddrAndKey(name) [StdCheatsSafe], 342
D3MCoreDeployScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 339
D3MCoreDeployScript.readReceipt(path, index) [StdCheatsSafe], 340
D3MCoreDeployScript.readReceipts(path) [StdCheatsSafe], 340
D3MCoreDeployScript.readTx1559(path, index) [StdCheatsSafe], 340
D3MCoreDeployScript.readTx1559s(path) [StdCheatsSafe], 340
D3MCoreDeployScript.run() X, 346
D3MCoreDeployScript.setChain(chainAlias, chain) [StdChains], 344
D3MCoreInitScript.assumeNoPrecompiles(addr, chainId) [StdCheatsSafe], 366
D3MCoreInitScript.modifier noGasMetering() [StdCheatsSafe], 363
D3MCoreInitScript.bound(x, min, max) [StdUtils], 364
D3MCoreInitScript.deployCode(what) [StdCheatsSafe], 370
D3MCoreInitScript.deployCode(what, args) [StdCheatsSafe], 369
D3MCoreInitScript.deployCode(what, args, val) [StdCheatsSafe], 370
D3MCoreInitScript.deployCode(what, val) [StdCheatsSafe], 370
D3MCoreInitScript.deriveRememberKey(mnemonic, index) [StdCheatsSafe], 371
D3MCoreInitScript.getChain(chainId) [StdChains], 371
D3MCoreInitScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 373
D3MCoreInitScript.isFork() [StdCheatsSafe], 371
D3MCoreInitScript.makeAddrAndKey(name) [StdCheatsSafe], 370
D3MCoreInitScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 367
D3MCoreInitScript.readReceipt(path, index) [StdCheatsSafe], 368
D3MCoreInitScript.readReceipts(path) [StdCheatsSafe], 368
D3MCoreInitScript.readTx1559(path, index) [StdCheatsSafe], 368
D3MCoreInitScript.readTx1559s(path) [StdCheatsSafe], 368
D3MCoreInitScript.run() X, 374
D3MCoreInitScript.setChain(chainAlias, chain) [StdChains], 372
D3MDeployScript.assumeNoPrecompiles(addr, chainId) [StdCheatsSafe], 392
D3MDeployScript.modifier noGasMetering() [StdCheatsSafe], 389
D3MDeployScript.bound(x, min, max) [StdUtils], 390
D3MDeployScript.deployCode(what) [StdCheatsSafe], 396
D3MDeployScript.deployCode(what, args) [StdCheatsSafe], 395
D3MDeployScript.deployCode(what, args, val) [StdCheatsSafe], 396
D3MDeployScript.deployCode(what, val) [StdCheatsSafe], 396
D3MDeployScript.deriveRememberKey(mnemonic, index) [StdCheatsSafe], 397
D3MDeployScript.getChain(chainId) [StdChains], 397

D3MDeployScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 399
 D3MDeployScript.isFork() [StdCheatsSafe], 397
 D3MDeployScript.makeAddrAndKey(name) [StdCheatsSafe], 396
 D3MDeployScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 393
 D3MDeployScript.readReceipt(path, index) [StdCheatsSafe], 394
 D3MDeployScript.readReceipts(path) [StdCheatsSafe], 394
 D3MDeployScript.readTx1559(path, index) [StdCheatsSafe], 394
 D3MDeployScript.readTx1559s(path) [StdCheatsSafe], 394
 D3MDeployScript.run() **X**, 400
 D3MDeployScript.setChain(chainAlias, chain) [StdChains], 398
 D3MInitScript.assumeNoPrecompiles(addr, chainId) [StdCheatsSafe], 412
 D3MInitScript.modifier noGasMetering() [StdCheatsSafe], 409
 D3MInitScript.bound(x, min, max) [StdUtils], 410
 D3MInitScript.deployCode(what) [StdCheatsSafe], 416
 D3MInitScript.deployCode(what, args) [StdCheatsSafe], 416
 D3MInitScript.deployCode(what, args, val) [StdCheatsSafe], 416
 D3MInitScript.deployCode(what, val) [StdCheatsSafe], 416
 D3MInitScript.deriveRememberKey(mnemonic, index) [StdCheatsSafe], 417
 D3MInitScript.getChain(chainId) [StdChains], 418
 D3MInitScript.getChainWithUpdatedRpcUrl(chainAlias, chain) [StdChains], 419
 D3MInitScript.isFork() [StdCheatsSafe], 417
 D3MInitScript.makeAddrAndKey(name) [StdCheatsSafe], 417
 D3MInitScript.readEIP1559ScriptArtifact(path) [StdCheatsSafe], 413
 D3MInitScript.readReceipt(path, index) [StdCheatsSafe], 414
 D3MInitScript.readReceipts(path) [StdCheatsSafe], 414
 D3MInitScript.readTx1559(path, index) [StdCheatsSafe], 414
 D3MInitScript.readTx1559s(path) [StdCheatsSafe], 414
 D3MInitScript.run() **X**, 420
 D3MInitScript.setChain(chainAlias, chain) [StdChains], 418
 ScriptTools.exportContract(label, addr), 715
 ScriptTools.exportContract(name, label, addr), 715
 ScriptTools.getRootChainId(), 713
 ScriptTools.loadConfig(), 714
 ScriptTools.loadConfig(name), 714
 ScriptTools.loadDependencies(), 715
 ScriptTools.loadDependencies(name), 714
 ScriptTools.readAddress(json, key, envKey), 717
 ScriptTools.readAddressArray(json, key, envKey), 718
 ScriptTools.readBool(json, key, envKey), 718
 ScriptTools.readBoolArray(json, key, envKey), 718
 ScriptTools.readBytes(json, key, envKey), 718
 ScriptTools.readBytes32(json, key, envKey), 717
 ScriptTools.readBytes32Array(json, key, envKey), 717
 ScriptTools.readByteArray(json, key, envKey), 718
 ScriptTools.readInput(name), 713
 ScriptTools.readInt(json, key, envKey), 717
 ScriptTools.readIntArray(json, key, envKey), 717
 ScriptTools.readOutput(name), 714
 ScriptTools.readOutput(name, timestamp), 713
 ScriptTools.readString(json, key, envKey), 717
 ScriptTools.readStringArray(json, key, envKey), 717
 ScriptTools.readUint(json, key, envKey), 716
 ScriptTools.readUintArray(json, key, envKey), 717

VM.ADDR

definition

GodMode.VM_ADDR: address, 767

read

GodMode.vm(), 767

VM.ADDRESS

definition

D3MCoreDeployScript.VM_ADDRESS [CommonBase] : address, 329

D3MCoreInitScript.VM_ADDRESS [CommonBase] : address, 357

D3MDeployScript.VM_ADDRESS [CommonBase] : address, 383


```

    D3MInitScript.VM_ADDRESS [CommonBase] : address, 403
vmSafe
  definition
    D3MCoreDeployScript.vmSafe [ScriptBase] : VmSafe, 329
    D3MCoreInitScript.vmSafe [ScriptBase] : VmSafe, 357
    D3MDeployScript.vmSafe [ScriptBase] : VmSafe, 383
    D3MInitScript.vmSafe [ScriptBase] : VmSafe, 403
void
  definition
    OSM.void\(\) X a, 287
Vow
  definition
    contract Vow, 269
vow
  call
    DssExecLib.addCollateralBase(_ilk, _gem, _join, _clip, _calc, _pip) X, 749
    DssExecLib.sendPaymentFromSurplusBuffer(_target, _amount) X, 751
    DssExecLib.setDebtAuctionDAIAmount(_amount) X, 739
    DssExecLib.setDebtAuctionDelay(_duration) X, 739
    DssExecLib.setDebtAuctionMKRAmount(_amount) X, 739
    DssExecLib.setSurplusAuctionAmount(_amount) X, 738
    DssExecLib.setSurplusBuffer(_amount) X, 738
  write
    Clipper.file(what, data) X a, 243
    D3MHub.file(what, data) X a, 321
    Dog.file(what, data) X a, 252
    DssPsm.constructor(gemJoin_, daiJoin_, vow_) X, 509
    End.file(what, data) X a, 298
    Flopper.cage() X a, 267
    Jug.file(what, data) X a, 275
    Pot.file(what, addr) X a, 278
    TeleportJoin.file(what, data) X a, 642
  definition
    Clipper.vow: address, 241
    D3MHub.vow: address, 319
    Dog.vow: VowLike, 251
    DssExecLib.vow(), 731
    DssPsm.vow: address, 509
    End.vow: VowLike, 295
    Flopper.vow: address, 265
    Jug.vow: address, 273
    Pot.vow: address, 277
    TeleportJoin.vow: address, 641
  read
    Clipper.kick(tab, lot, usr, kpr) X a, 244
    Clipper.redo(id, kpr) X, 245
    Clipper.take(id, amt, max, who, data) X, 246
    D3MHub.cull(ilk) X, 326
    D3MHub._exec(ilk, _pool, Art, lineWad), 322
    D3MHub.uncull(ilk) X, 326
    D3MHub._wipe(ilk, _pool, urn), 322
    Dog.bark(ilk, urn, kpr) X, 253
    DssPsm.buyGem(usr, gemAmt) X, 511
    DssPsm.sellGem(usr, gemAmt) X, 510
    End.cage() X a, 299
    End.free(ilk) X, 300
    End.pack(wad) X, 301
    End.skim(ilk, urn) X, 300
    End.skip(ilk, id) X, 299
    End.snip(ilk, id) X, 299
    End.thaw() X, 300
    Flopper.yank(id) X, 267
    Jug.drip(ilk) X, 275

```

Pot.drip() **X**, 279

TeleportJoin.mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643

W

WAD

definition

ClipperMom.WAD: [uint256](#), 257
 Clipper.WAD: [uint256](#), 241
 D3MCompoundPlan.WAD: [uint256](#), 315
 D3MCompoundPool.WAD: [uint256](#), 377
 D3MHub.WAD: [uint256](#), 319
 D3MOracle.WAD: [uint256](#), 353
 Dog.WAD: [uint256](#), 251
 DssExecLib.WAD: [uint256](#), 729
 DssPsm.WAD: [uint256](#), 509
 End.WAD: [uint256](#), 295
 ESM.WAD: [uint256](#), 303
 MCD.WAD: [uint256](#), 719
 TeleportJoin.WAD: [uint256](#), 641
 TeleportLinearFee.WAD: [uint256](#), 647

read

Clipper.wmul(x, y), 244
 D3MCompoundPlan._wdiv(x, y), 316
 D3MCompoundPlan._wmul(x, y), 315
 D3MCompoundPool._wdiv(x, y), 378
 D3MCompoundPool._wmul(x, y), 378
 D3MHub._exec(ilk, _pool, Art, lineWad), 322
 D3MOracle.peek(), 354
 Dog.bark(ilk, urn, kpr) [X](#), 253
 Dog.file(ilk, what, [data](#)) [X](#) [a](#), 253
 DssExecLib.decreaseGlobalDebtCeiling(_amount) [X](#), 737
 DssExecLib.decreaseIlkDebtCeiling(_ilk, _amount, _global) [X](#), 742
 DssExecLib.increaseGlobalDebtCeiling(_amount) [X](#), 737
 DssExecLib.increaseIlkDebtCeiling(_ilk, _amount, _global) [X](#), 741
 DssExecLib.sendPaymentFromSurplusBuffer(_target, _amount) [X](#), 751
 DssExecLib.setDAIReferenceValue(_value) [X](#), 741
 DssExecLib.setDebtAuctionDAIAmount(_amount) [X](#), 739
 DssExecLib.setDebtAuctionMKRAmount(_amount) [X](#), 739
 DssExecLib.setDebtAuctionMKRIncreaseRate(_pct_bps) [X](#), 740
 DssExecLib.setGlobalDebtCeiling(_amount) [X](#), 737
 DssExecLib.setIlkAutoLineParameters(_ilk, _amount, _gap, _ttl) [X](#), 742
 DssExecLib.setIlkDebtCeiling(_ilk, _amount) [X](#), 741
 DssExecLib.setIlkLiquidationPenalty(_ilk, _pct_bps) [X](#), 743
 DssExecLib.setIlkMaxLiquidationAmount(_ilk, _amount) [X](#), 744
 DssExecLib.setIlkMinVaultAmount(_ilk, _amount) [X](#), 743
 DssExecLib.setKeeperIncentiveFlatRate(_ilk, _amount) [X](#), 745
 DssExecLib.setMaxTotalDAILiquidationAmount(_amount) [X](#), 740
 DssExecLib.setMaxTotalDAILiquidationAmountLEGACY(_amount) [X](#), 740
 DssExecLib.setMinDebtAuctionBidIncrease(_pct_bps) [X](#), 739
 DssExecLib.setMinSurplusAuctionBidIncrease(_pct_bps) [X](#), 738
 DssExecLib.setRWAILkDebtCeiling(_ilk, _ceiling, _price) [X](#), 742
 DssExecLib.setSurplusAuctionAmount(_amount) [X](#), 738
 DssExecLib.setSurplusBuffer(_amount) [X](#), 738
 DssExecLib.wdiv(x, y), 730
 DssPsm.buyGem(usr, gemAmt) [X](#), 511
 DssPsm.sellGem(usr, gemAmt) [X](#), 510
 End.wdiv(x, y), 298
 ESM.file(what, [data](#)) [X](#) [a](#), 304
 MCD.initIlk(dss, ilk), 720
 MCD.initIlk(dss, ilk, join), 720
 TeleportJoin._mint(teleportGUID, hashGUID, maxFeePercentage, operatorFee), 643
 TeleportLinearFee._getFee(guid, , , , amtToTake), 647

WAD_BPS

definition

TrustedRelay.WAD_BPS: [uint256](#), 661

read

TrustedRelay.relay(teleportGUID, signatures, maxFeePercentage, gasFee, expiry, v, r, s) [X](#), 663

wait

write

Cure.file(what, data) [X](#) [a](#), 292

End.file(what, data) [X](#) [a](#), 298

Vow.file(what, data) [X](#) [a](#), 270

definition

Cure.wait: [uint256](#), 291

End.wait: [uint256](#), 295

Vow.wait: [uint256](#), 269

read

Cure.cage() [X](#) [a](#), 293

End.thaw() [X](#), 300

Vow.flog(era) [X](#), 270

wards

call

GodMode.setWard(base, target, val), 767

IlkRegistry.add(adapter) [X](#), 668

IlkRegistry.constructor(vat_, dog_, cat_, spot_) [X](#), 668

ScriptTools.switchOwner(base, deployer, newOwner), 716

write

AuthGemJoin5.constructor(vat_, ilk_, gem_) [X](#), 503

AuthGemJoin5.deny(usr) [X](#) [a](#), 503

AuthGemJoin5.rely(usr) [X](#) [a](#), 503

AuthGemJoin8.constructor(vat_, ilk_, gem_) [X](#), 505

AuthGemJoin8.deny(usr) [X](#) [a](#), 505

AuthGemJoin8.rely(usr) [X](#) [a](#), 505

AuthGemJoin.constructor(vat_, ilk_, gem_) [X](#), 447, 507

AuthGemJoin.deny(usr) [X](#) [a](#), 447, 507

AuthGemJoin.rely(usr) [X](#) [a](#), 447, 507

BasicRelay.constructor(_oracleAuth, _daiJoin) [X](#), 657

BasicRelay.deny(usr) [X](#) [a](#), 657

BasicRelay.rely(usr) [X](#) [a](#), 657

ChainLog.constructor() [X](#), 709

ChainLog.deny(usr) [X](#) [a](#), 709

ChainLog.rely(usr) [X](#) [a](#), 709

Clipper.constructor(vat_, spotter_, dog_, ilk_) [X](#), 243

Clipper.deny(usr) [X](#) [a](#), 242

Clipper.rely(usr) [X](#) [a](#), 242

Cure.constructor() [X](#), 292

Cure.deny(usr) [X](#) [a](#), 292

Cure.rely(usr) [X](#) [a](#), 292

D3MAavePlan.constructor(dai_, pool_) [X](#), 311

D3MAavePlan.deny(usr) [X](#) [a](#), 312

D3MAavePlan.rely(usr) [X](#) [a](#), 312

D3MAavePool.constructor(ilk_, hub_, dai_, pool_) [X](#), 349

D3MAavePool.deny(usr) [X](#) [a](#), 350

D3MAavePool.rely(usr) [X](#) [a](#), 350

D3MCompoundPlan.constructor(cDai_) [X](#), 315

D3MCompoundPlan.deny(usr) [X](#) [a](#), 316

D3MCompoundPlan.rely(usr) [X](#) [a](#), 316

D3MCompoundPool.constructor(ilk_, hub_, cDai_) [X](#), 377

D3MCompoundPool.deny(usr) [X](#) [a](#), 378

D3MCompoundPool.rely(usr) [X](#) [a](#), 378

D3MHub.constructor(daiJoin_) [X](#), 320

D3MHub.deny(usr) [X](#) [a](#), 321

D3MHub.rely(usr) [X](#) [a](#), 321

D3MOracle.constructor(vat_, ilk_) [X](#), 353

D3MOracle.deny(usr) [X](#) [a](#), 353

D3MOracle.rely(usr) [X](#) [a](#), 353

Dai.constructor() [X](#), 588, 614

Dai.constructor(chainId_) [X](#), 224

Dai.deny(guy) [X](#) [a](#), 223
Dai.deny(usr) [X](#) [a](#), 587, 613
DaiJoin.constructor(vat_, dai_) [X](#), 425
DaiJoin.deny(usr) [X](#) [a](#), 425
DaiJoin.rely(usr) [X](#) [a](#), 425
Dai.rely(guy) [X](#) [a](#), 223
Dai.rely(usr) [X](#) [a](#), 587, 613
Dog.constructor(vat_) [X](#), 252
Dog.deny(usr) [X](#) [a](#), 252
Dog.rely(usr) [X](#) [a](#), 252
DssAutoLine.constructor(vat_) [X](#), 705
DssAutoLine.deny(usr) [X](#) [a](#), 707
DssAutoLine.rely(usr) [X](#) [a](#), 707
DssPsm.constructor(gemJoin_, daiJoin_, vow_) [X](#), 509
DssPsm.deny(usr) [X](#) [a](#), 509
DssPsm.rely(usr) [X](#) [a](#), 509
DssVestMintable.constructor() [DssVest] [X](#), 677
DssVestMintable.deny(_usr) [DssVest] [X](#) [a](#), 677
DssVestMintable.rely(_usr) [DssVest] [X](#) [a](#), 677
DssVestSuckable.constructor() [DssVest] [X](#), 687
DssVestSuckable.deny(_usr) [DssVest] [X](#) [a](#), 687
DssVestSuckable.rely(_usr) [DssVest] [X](#) [a](#), 687
DssVestTransferrable.constructor() [DssVest] [X](#), 697
DssVestTransferrable.deny(_usr) [DssVest] [X](#) [a](#), 697
DssVestTransferrable.rely(_usr) [DssVest] [X](#) [a](#), 697
End.constructor() [X](#), 297
End.deny(usr) [X](#) [a](#), 297
End.rely(usr) [X](#) [a](#), 297
ESM.constructor(gem_, end_, proxy_, min_) [X](#), 303
ESM.deny(usr) [X](#) [a](#), 304
ESM.rely(usr) [X](#) [a](#), 304
ExponentialDecrease.constructor() [X](#), 239
ExponentialDecrease.deny(usr) [X](#) [a](#), 239
ExponentialDecrease.rely(usr) [X](#) [a](#), 239
Flapper.constructor(vat_, gem_) [X](#), 262
Flapper.deny(usr) [X](#) [a](#), 262
Flapper.rely(usr) [X](#) [a](#), 262
Flopper.constructor(vat_, gem_) [X](#), 266
Flopper.deny(usr) [X](#) [a](#), 266
Flopper.rely(usr) [X](#) [a](#), 265
GemJoin2.constructor(vat_, ilk_, gem_) [X](#), 427
GemJoin2.deny(usr) [X](#) [a](#), 427
GemJoin2.rely(usr) [X](#) [a](#), 427
GemJoin3.constructor(vat_, ilk_, gem_, decimals) [X](#), 429
GemJoin3.deny(usr) [X](#) [a](#), 429
GemJoin3.rely(usr) [X](#) [a](#), 429
GemJoin4.constructor(vat_, ilk_, gem_) [X](#), 433
GemJoin4.deny(usr) [X](#) [a](#), 433
GemJoin4.rely(usr) [X](#) [a](#), 433
GemJoin5.constructor(vat_, ilk_, gem_) [X](#), 435
GemJoin5.deny(usr) [X](#) [a](#), 435
GemJoin5.rely(usr) [X](#) [a](#), 435
GemJoin6.constructor(vat_, ilk_, gem_) [X](#), 437
GemJoin6.deny(usr) [X](#) [a](#), 437
GemJoin6.rely(usr) [X](#) [a](#), 437
GemJoin7.constructor(vat_, ilk_, gem_) [X](#), 439
GemJoin7.deny(usr) [X](#) [a](#), 439
GemJoin7.rely(usr) [X](#) [a](#), 439
GemJoin8.constructor(vat_, ilk_, gem_) [X](#), 441
GemJoin8.deny(usr) [X](#) [a](#), 441
GemJoin8.rely(usr) [X](#) [a](#), 441
GemJoin9.constructor(vat_, ilk_, gem_) [X](#), 443
GemJoin9.deny(usr) [X](#) [a](#), 443

GemJoin9.rely(usr) [X](#) [a](#), 443
 GemJoin.constructor(vat_, ilk_, gem_) [X](#), 424
 GemJoin.deny(usr) [X](#) [a](#), 424
 GemJoin.rely(usr) [X](#) [a](#), 424
 IlkRegistry.constructor(vat_, dog_, cat_, spot_) [X](#), 668
 IlkRegistry.deny(usr) [X](#) [a](#), 667
 IlkRegistry.rely(usr) [X](#) [a](#), 667
 Jug.constructor(vat_) [X](#), 273
 Jug.deny(usr) [X](#) [a](#), 273
 Jug.rely(usr) [X](#) [a](#), 273
 L1DAIBridge.constructor(_starkNet, _dai, _l2Dai, _escrow, _l2DaiBridge) [X](#), 618
 L1DAIBridge.deny(usr) [X](#) [a](#), 618
 L1DAIBridge.rely(usr) [X](#) [a](#), 618
 L1DaiGateway.constructor(_l2Counterpart, _l1Router, _inbox, _l1Dai, _l2Dai, _l1Escrow) [X](#), 571
 L1DaiGateway.deny(usr) [X](#) [a](#), 571
 L1DaiGateway.rely(usr) [X](#) [a](#), 570
 L1DAITokenBridge.constructor(_l1Token, _l2DAITokenBridge, _l2Token, _l1messenger, _escrow) [X](#), 593
 L1DAITokenBridge.deny(usr) [X](#) [a](#), 593
 L1DAITokenBridge.rely(usr) [X](#) [a](#), 593
 L1Escrow.constructor() [X](#), 575, 597, 635
 L1Escrow.deny(usr) [X](#) [a](#), 575, 597, 635
 L1Escrow.rely(usr) [X](#) [a](#), 575, 597, 635
 L1GovernanceRelay.constructor(_inbox, _l2GovernanceRelay) [X](#), 579
 L1GovernanceRelay.constructor(_l2GovernanceRelay, _l1messenger) [X](#), 601
 L1GovernanceRelay.constructor(_starkNet, _l2GovernanceRelay) [X](#), 621
 L1GovernanceRelay.deny(usr) [X](#) [a](#), 578, 601, 621
 L1GovernanceRelay.rely(usr) [X](#) [a](#), 578, 601, 621
 L2DaiGateway.constructor(_l1Counterpart, _l2Router, _l1Dai, _l2Dai) [X](#), 582
 L2DaiGateway.deny(usr) [X](#) [a](#), 582
 L2DaiGateway.rely(usr) [X](#) [a](#), 582
 L2DAITokenBridge.constructor(_l2CrossDomainMessenger, _l2Token, _l1Token, _l1DAITokenBridge) [X](#), 605
 L2DAITokenBridge.deny(usr) [X](#) [a](#), 605
 L2DAITokenBridge.rely(usr) [X](#) [a](#), 605
 LinearDecrease.constructor() [X](#), 235
 LinearDecrease.deny(usr) [X](#) [a](#), 235
 LinearDecrease.rely(usr) [X](#) [a](#), 235
 ManagedGemJoin.constructor(vat_, ilk_, gem_) [X](#), 449
 ManagedGemJoin.deny(usr) [X](#) [a](#), 449
 ManagedGemJoin.rely(usr) [X](#) [a](#), 449
 Median.constructor() [X](#), 282
 Median.deny(usr) [X](#) [a](#), 282
 Median.rely(usr) [X](#) [a](#), 282
 OSM.constructor(src_) [X](#), 286
 OSM.deny(usr) [X](#) [a](#), 286
 OSM.rely(usr) [X](#) [a](#), 286
 Pot.constructor(vat_) [X](#), 277
 Pot.deny(guy) [X](#) [a](#), 277
 Pot.rely(guy) [X](#) [a](#), 277
 Spotter.constructor(vat_) [X](#), 227
 Spotter.deny(guy) [X](#) [a](#), 227
 Spotter.rely(guy) [X](#) [a](#), 227
 StairstepExponentialDecrease.constructor() [X](#), 237
 StairstepExponentialDecrease.deny(usr) [X](#) [a](#), 237
 StairstepExponentialDecrease.rely(usr) [X](#) [a](#), 237
 TeleportJoin.constructor(vat_, daiJoin_, ilk_, domain_) [X](#), 641
 TeleportJoin.deny(usr) [X](#) [a](#), 642
 TeleportJoin.rely(usr) [X](#) [a](#), 642
 TeleportOracleAuth.constructor(teleportJoin_) [X](#), 649
 TeleportOracleAuth.deny(usr) [X](#) [a](#), 649
 TeleportOracleAuth.rely(usr) [X](#) [a](#), 649

TeleportRouter.constructor(dai_) X, 653
 TeleportRouter.deny(usr) X a, 653
 TeleportRouter.rely(usr) X a, 653
 TrustedRelay.constructor(_oracleAuth, _daiJoin, _ethPriceOracle) X, 661
 TrustedRelay.deny(usr) X a, 662
 TrustedRelay.rely(usr) X a, 662
 Vat.constructor() X, 230
 Vat.deny(usr) X a, 229
 Vat.rely(usr) X a, 229
 Vow.constructor(vat_, flapper_, flopper_) X, 269
 Vow.deny(usr) X a, 269
 Vow.rely(usr) X a, 269

definition

AuthGemJoin5.wards: mapping(address => uint256), 503
 AuthGemJoin8.wards: mapping(address => uint256), 505
 AuthGemJoin.wards: mapping(address => uint256), 447, 507
 BasicRelay.wards: mapping(address => uint256), 657
 ChainLog.wards: mapping(address => uint256), 709
 Clipper.wards: mapping(address => uint256), 241
 Cure.wards: mapping(address => uint256), 291
 D3MAavePlan.wards: mapping(address => uint256), 311
 D3MAavePool.wards: mapping(address => uint256), 349
 D3MCompoundPlan.wards: mapping(address => uint256), 315
 D3MCompoundPool.wards: mapping(address => uint256), 377
 D3MHub.wards: mapping(address => uint256), 319
 D3MOracle.wards: mapping(address => uint256), 353
 DaiJoin.wards: mapping(address => uint256), 425
 Dai.wards: mapping(address => uint256), 223, 587, 613
 Dog.wards: mapping(address => uint256), 251
 DssAutoLine.wards: mapping(address => uint256), 705
 DssPsm.wards: mapping(address => uint256), 509
 DssVestMintable.wards [DssVest] : mapping(address => uint256), 675
 DssVestSuckable.wards [DssVest] : mapping(address => uint256), 685
 DssVestTransferrable.wards [DssVest] : mapping(address => uint256), 695
 End.wards: mapping(address => uint256), 295
 ESM.wards: mapping(address => uint256), 303
 ExponentialDecrease.wards: mapping(address => uint256), 239
 Flapper.wards: mapping(address => uint256), 261
 Flopper.wards: mapping(address => uint256), 265
 GemJoin2.wards: mapping(address => uint256), 427
 GemJoin3.wards: mapping(address => uint256), 429
 GemJoin4.wards: mapping(address => uint256), 433
 GemJoin5.wards: mapping(address => uint256), 435
 GemJoin6.wards: mapping(address => uint256), 437
 GemJoin7.wards: mapping(address => uint256), 439
 GemJoin8.wards: mapping(address => uint256), 441
 GemJoin9.wards: mapping(address => uint256), 443
 GemJoin.wards: mapping(address => uint256), 423
 IlkRegistry.wards: mapping(address => uint256), 667
 Jug.wards: mapping(address => uint256), 273
 L1DAIBridge.wards: mapping(address => uint256), 617
 L1DaiGateway.wards: mapping(address => uint256), 569
 L1DAITokenBridge.wards: mapping(address => uint256), 591
 L1Escrow.wards: mapping(address => uint256), 575, 597, 635
 L1GovernanceRelay.wards: mapping(address => uint256), 577, 599, 621
 L2DaiGateway.wards: mapping(address => uint256), 581
 L2DAITokenBridge.wards: mapping(address => uint256), 603
 LinearDecrease.wards: mapping(address => uint256), 235
 ManagedGemJoin.wards: mapping(address => uint256), 449
 Median.wards: mapping(address => uint256), 281
 OSM.wards: mapping(address => uint256), 285
 Pot.wards: mapping(address => uint256), 277
 Spotter.wards: mapping(address => uint256), 227

StairstepExponentialDecrease.wards: `mapping(address => uint256)`, 237
 TeleportJoin.wards: `mapping(address => uint256)`, 641
 TeleportOracleAuth.wards: `mapping(address => uint256)`, 649
 TeleportRouter.wards: `mapping(address => uint256)`, 653
 TrustedRelay.wards: `mapping(address => uint256)`, 661
 Vat.wards: `mapping(address => uint256)`, 229
 Vow.wards: `mapping(address => uint256)`, 269

read

AuthGemJoin5.modifier `auth()`, 503
 AuthGemJoin8.modifier `auth()`, 505
 AuthGemJoin.modifier `auth()`, 447, 507
 BasicRelay.modifier `auth()`, 657
 ChainLog.modifier `auth()`, 709
 Clipper.modifier `auth()`, 242
 Cure.modifier `auth()`, 291
 D3MAavePlan.modifier `auth()`, 311
 D3MAavePlan.disable() **X**, 314
 D3MAavePool.modifier `auth()`, 349
 D3MCompoundPlan.modifier `auth()`, 315
 D3MCompoundPlan.disable() **X**, 317
 D3MCompoundPool.modifier `auth()`, 377
 D3MHub.modifier `auth()`, 320
 D3MHub.cull(`ilk`) **X**, 326
 D3MOracle.modifier `auth()`, 353
 Dai.modifier `auth()`, 223, 587, 613
 Dai.burn(`from`, `value`) **X**, 590, 616
 DaiJoin.modifier `auth()`, 425
 Dog.modifier `auth()`, 251
 DssAutoLine.modifier `auth()`, 705
 DssPsm.modifier `auth()`, 509
 DssVestMintable.modifier `auth()` [`DssVest`], 676
 DssVestMintable.restrict(`_id`) [`DssVest`] **X**, 681
 DssVestMintable.unrestrict(`_id`) [`DssVest`] **X**, 681
 DssVestMintable.yank(`_id`, `_end`) [`DssVest`], 682
 DssVestSuckable.modifier `auth()` [`DssVest`], 686
 DssVestSuckable.restrict(`_id`) [`DssVest`] **X**, 690
 DssVestSuckable.unrestrict(`_id`) [`DssVest`] **X**, 691
 DssVestSuckable.yank(`_id`, `_end`) [`DssVest`], 691
 DssVestTransferrable.modifier `auth()` [`DssVest`], 696
 DssVestTransferrable.restrict(`_id`) [`DssVest`] **X**, 701
 DssVestTransferrable.unrestrict(`_id`) [`DssVest`] **X**, 701
 DssVestTransferrable.yank(`_id`, `_end`) [`DssVest`], 701
 End.modifier `auth()`, 297
 ESM.modifier `auth()`, 303
 ExponentialDecrease.modifier `auth()`, 239
 Flapper.modifier `auth()`, 262
 Flopper.modifier `auth()`, 265
 GemJoin2.modifier `auth()`, 427
 GemJoin3.modifier `auth()`, 429
 GemJoin4.modifier `auth()`, 433
 GemJoin5.modifier `auth()`, 435
 GemJoin6.modifier `auth()`, 437
 GemJoin7.modifier `auth()`, 439
 GemJoin8.modifier `auth()`, 441
 GemJoin9.modifier `auth()`, 443
 GemJoin.modifier `auth()`, 423
 IlkRegistry.modifier `auth()`, 667
 Jug.modifier `auth()`, 273
 L1DAIBridge.modifier `auth()`, 618
 L1DaiGateway.modifier `auth()`, 569
 L1DAITokenBridge.modifier `auth()`, 592
 L1Escrow.modifier `auth()`, 575, 597, 635
 L1GovernanceRelay.modifier `auth()`, 577, 600, 621

- L2DaiGateway.modifier auth(), 581
- L2DAITokenBridge.modifier auth(), 604
- LinearDecrease.modifier auth(), 235
- ManagedGemJoin.modifier auth(), 449
- Median.modifier auth(), 282
- OSM.modifier auth(), 286
- Pot.modifier auth(), 277
- Spotter.modifier auth(), 227
- StairstepExponentialDecrease.modifier auth(), 237
- TeleportJoin.modifier auth(), 641
- TeleportOracleAuth.modifier auth(), 649
- TeleportRouter.modifier auth(), 653
- TrustedRelay.modifier auth(), 661
- Vat.modifier auth(), 229
- Vow.modifier auth(), 269

wat

- definition
 - Median.wat: bytes32, 281
- read
 - Median.recover(val_, age_, v, r, s), 283

WBTC

- definition
 - contract WBTC, 495

wdiv

- call
 - D3MCompoundPlan._calculateTargetSupply(targetInterestRate, borrows), 316
 - D3MCompoundPool.deposit(wad) X, 378
 - DssExecLib.setDebtAuctionMKRIncreaseRate(_pct_bps) X, 740
 - DssExecLib.setIlkLiquidationPenalty(_ilk, _pct_bps) X, 743
 - DssExecLib.setKeeperIncentivePercent(_ilk, _pct_bps) X, 745
 - DssExecLib.setMinDebtAuctionBidIncrease(_pct_bps) X, 739
 - DssExecLib.setMinSurplusAuctionBidIncrease(_pct_bps) X, 738
 - End.cage(ilk) X, 299
- definition
 - D3MCompoundPlan._wdiv(x, y), 316
 - D3MCompoundPool._wdiv(x, y), 378
 - DssExecLib.wdiv(x, y), 730
 - End.wdiv(x, y), 298

WETH

- definition
 - contract WETH, 515

weth

- write
 - CurveLpTokenUniv3Callee.constructor(uniV3Router_, daiJoin_, weth_) X, 551
 - WstETHCurveUniv3Callee.constructor(curvePool_, uniV3Router_, daiJoin_, weth_) X, 563
- definition
 - CurveLpTokenUniv3Callee.weth: address, 551
 - WstETHCurveUniv3Callee.weth: address, 563
- read
 - CurveLpTokenUniv3Callee.clipperCall(sender, owe, slice, data) X, 552
 - WstETHCurveUniv3Callee.clipperCall(sender, owe, slice, data) X, 564

WETH9_

- definition
 - contract WETH9_, 517

when

- write
 - Cure.cage() X a, 293
 - End.cage() X a, 299
- definition
 - Cure.when: uint256, 291
 - End.when: uint256, 295
- read

Cure.tell(), 292
 End.thaw() [X](#), 300
 whenOpen
 call
 L1DAIBridge.deposit(amount, l2Recipient) [X](#), 619
 L1DAIBridge.forceWithdrawal(amount, l2Sender) [X](#), 620
 L1DAIBridge.setCeiling(_ceiling) [X](#) [a](#), 619
 L1DAIBridge.setMaxDeposit(_maxDeposit) [X](#) [a](#), 619
 definition
 L1DAIBridge.modifier whenOpen(), 618
 whitelistOracleMedians
 call
 DssExecLib.addNewCollateral(co) [X](#), 750
 definition
 DssExecLib.whitelistOracleMedians(_oracle) [X](#), 747
 wipe
 call
 D3MHub.exec(ilk) [X](#), 324
 DssProxyActions.safeWipe(manager, daiJoin, cdp, wad, owner) [X](#), 537
 definition
 D3MHub._wipe(ilk, _pool, urn), 322
 DssProxyActions.wipe(manager, daiJoin, cdp, wad) [X](#), 537
 wipeAll
 call
 DssProxyActions.safeWipeAll(manager, daiJoin, cdp, owner) [X](#), 538
 definition
 DssProxyActions.wipeAll(manager, daiJoin, cdp) [X](#), 537
 wipeAllAndFreeETH
 definition
 DssProxyActions.wipeAllAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC) [X](#), 540
 wipeAllAndFreeGem
 definition
 DssProxyActions.wipeAllAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC) [X](#), 541
 wipeAndFreeETH
 definition
 DssProxyActions.wipeAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC, wadD) [X](#), 540
 wipeAndFreeGem
 definition
 DssProxyActions.wipeAndFreeGem(manager, gemJoin, daiJoin, cdp, amtC, wadD) [X](#), 541
 wish
 call
 Vat.flux(ilk, src, dst, wad) [X](#), 231
 Vat.fork(ilk, src, dst, dink, dart) [X](#), 233
 Vat.frob(i, u, v, w, dink, dart) [X](#), 232
 Vat.move(src, dst, rad) [X](#), 232
 definition
 Vat.wish(bit, usr), 230
 withdraw
 call
 D3MAavePool.withdraw(wad) [X](#), 351
 D3MHub._exec(ilk, _pool, Art, lineWad), 322
 D3MHub._wipe(ilk, _pool, urn), 322
 DssProxyActionsEnd.cashETH(ethJoin, end, ilk, wad) [X](#), 544
 DssProxyActionsEnd.freeETH(manager, ethJoin, end, cdp) [X](#), 544
 DssProxyActions.exitETH(manager, ethJoin, cdp, wad) [X](#), 536
 DssProxyActions.freeETH(manager, ethJoin, cdp, wad) [X](#), 535
 DssProxyActions.wipeAllAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC) [X](#), 540
 DssProxyActions.wipeAndFreeETH(manager, ethJoin, daiJoin, cdp, wadC, wadD) [X](#), 540
 definition
 D3MAavePool.withdraw(wad) [X](#), 351
 D3MCompoundPool.withdraw(wad) [X](#), 379
 L1DAIBridge.withdraw(amount, l1Recipient) [X](#), 619
 L2DAITokenBridge.withdraw(_l2Token, _amount, _l1Gas, _data) [X](#), 605

- WETH9_.withdraw(wad) [X](#), 517
- withdrawTo
 - [definition](#)
 - L2DAITokenBridge.withdrawTo(_l2Token, _to, _amount, _l1Gas, _data) [X](#), 606
- _wmul
 - [call](#)
 - Clipper.kick(tab, lot, usr, kpr) [X](#) [a](#), 244
 - Clipper.redo(id, kpr) [X](#), 245
 - Clipper.upchost() [X](#), 248
 - D3MCompoundPlan.calculateTargetSupply(targetInterestRate, borrows), 316
 - D3MCompoundPool.assetBalance(), 379
 - [definition](#)
 - Clipper.wmul(x, y), 244
 - D3MCompoundPlan.wmul(x, y), 315
 - D3MCompoundPool.wmul(x, y), 378
- writeJson
 - [call](#)
 - ScriptTools.exportContract(name, label, addr), 715
- WSTETH
 - [definition](#)
 - [contract](#) WSTETH, 497
- WstETHCurveUniv3Callee
 - [definition](#)
 - [contract](#) WstETHCurveUniv3Callee, 563

X

xDomainMessageSender

call

```

L1DAITokenBridge.modifier onlyFromCrossDomainAccount(_sourceDomainAccount)
    [OVM_CrossDomainEnabled], 592
L1GovernanceRelay.modifier onlyFromCrossDomainAccount(_sourceDomainAccount)
    [OVM_CrossDomainEnabled], 599
L2DAITokenBridge.modifier onlyFromCrossDomainAccount(_sourceDomainAccount)
    [OVM_CrossDomainEnabled], 603
L2GovernanceRelay.modifier onlyFromCrossDomainAccount(_sourceDomainAccount)
    [OVM_CrossDomainEnabled], 609

```

xlip

call

```

DssExecLib.clip(_ilk), 732
DssExecLib.flip(_ilk), 732

```

write

```

IlkRegistry.file(ilk, what, data) X a, 669

```

definition

```

IlkRegistry.xlip(ilk), 672

```

read

```

IlkRegistry.xlip(ilk), 672

```

Y**_yank****call**

DssVestMintable.yank(_id) [DssVest] [X](#), 681
 DssVestMintable.yank(_id, _end) [DssVest] [X](#), 681
 DssVestSuckable.yank(_id) [DssVest] [X](#), 691
 DssVestSuckable.yank(_id, _end) [DssVest] [X](#), 691
 DssVestTransferrable.yank(_id) [DssVest] [X](#), 701
 DssVestTransferrable.yank(_id, _end) [DssVest] [X](#), 701
 End.skip(ilk, id) [X](#), 299
 End.snip(ilk, id) [X](#), 299

definition

Clipper.yank(id) [X](#) [a](#), 249
 DssVestMintable.yank(_id) [DssVest] [X](#), 681
 DssVestMintable._yank(_id, _end) [DssVest], 682
 DssVestMintable.yank(_id, _end) [DssVest] [X](#), 681
 DssVestSuckable.yank(_id) [DssVest] [X](#), 691
 DssVestSuckable._yank(_id, _end) [DssVest], 691
 DssVestSuckable.yank(_id, _end) [DssVest] [X](#), 691
 DssVestTransferrable.yank(_id) [DssVest] [X](#), 701
 DssVestTransferrable._yank(_id, _end) [DssVest], 701
 DssVestTransferrable.yank(_id, _end) [DssVest] [X](#), 701
 Flapper.yank(id) [X](#), 264
 Flopper.yank(id) [X](#), 267

YFI**definition**

contract YFI, 499

Z

ZRX

definition

`contract` ZRX, 501

zzz

write

`OSM.poke()` `X`, 287

definition

`OSM.zzz:` `uint64`, 285

read

`OSM.pass()`, 287

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 multiplies 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 continuous (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-
// ↪ ild 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 calculation 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;
}
```



```
// 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)
            }
        }
    }
}
```


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 s1 = uint8(uint256(signer) >> 152);
        require((bloom >> s1) % 2 == 0, "Median/oracle-already-signed");
        bloom += uint256(2) ** s1;
    }

    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,
            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 -
    ↪ 0 - 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 -
    ↪ 0 - 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 utilization
    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 illegal
    ↪ 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
// ↪ permissionlessly 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;
}
```




```

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  
    ↪ 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 adapted from Solmate implementation (https://github.com/Rari-Capital/solmate/blob/main/src/Utils/LibRLP.sol)
↳ 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]


```

        vm.assume(addr < address(0x030000000000000000000000000000000000000000000000000000000000000000)
        ↪ || addr > address(0x0300000000000000000000000000000000000000000000000000000000000000Ff
        ↪ ));
    }
    // 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 = rawToConvertedEIP1559s(rawArtifact.
    ↪ transactions);
    return artifact;
}

```

8.4.32 rawToConvertedEIP1559s(rawTxs) [StdCheatsSafe]

```

function rawToConvertedEIP1559s(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] = rawToConvertedEIP1559(rawTxs[i]);
    }
    return txs;
}

```

8.4.33 rawToConvertedEIP1559(rawTx) [StdCheatsSafe]

```

function rawToConvertedEIP1559(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
    ↪ x000000000000000000000000636F6e736F6c652e6c6f67;
    // 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 =
        11579208923731619542357098500868790785326998466564056403945758400791312963993
        ↪

    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
    ↪ x000000000000000000000000636F6e736F6c652e6c6f67;
```

```

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 adapted 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]


```

        vm.assume(addr < address(0x030000000000000000000000000000000000000000000000000000000000000000)
        ↪ || addr > address(0x030000000000000000000000000000000000000000000000000000000000000000Ff
        ↪ ));
    }
    // 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 rawToConvertedEIP1559s(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 rawToConvertedEIP1559(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
    ↪ x000000000000000000000000636F6e736F6c652e6c6667;
    // 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 =
        11579208923731619542357098500868790785326998466564056403945758400791312963993
        ↪

    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
    ↪ x000000000000000000000000636F6e736F6c652e6c6667;
```

```

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 adapted 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]


```

        vm.assume(addr < address(0x0300000000000000000000000000000000000000000000000000000000000000)
        ↪ || addr > address(0x0300000000000000000000000000000000000000000000000000000000000000Ff
        ↪ ));
    }
    // 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(rawTx) [StdCheatsSafe]

```

function rawToConvertedEIPTx1559s(RawTx1559[] memory rawTx) internal pure
    ↪ virtual returns (Tx1559[] memory) {
    Tx1559[] memory txs = new Tx1559[](rawTx.length);
    for (uint256 i; i < rawTx.length; i++) {
        txs[i] = rawToConvertedEIPTx1559(rawTx[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 rawToConvertedEIP1559s(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 rawToConvertedEIP1559(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
    ↪ x000000000000000000000000636F6e736F6c652e6c6667;
    // 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 =
        11579208923731619542357098500868790785326998466564056403945758400791312963993
        ↪

    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
    ↪ x0000000000000000000000000000000000000000000000000000000000000000;

    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 adapted 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));
}

```



```

    } 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(0x0100000000000000000000000000000000)
        ↪ || addr > address(0x01000000000000000000000000000000ff
        ↪ ));
        vm.assume(addr < address(0x0200000000000000000000000000000000)
        ↪ || addr > address(0x02000000000000000000000000000000FF
        ↪ ));
        vm.assume(addr < address(0x0300000000000000000000000000000000)
        ↪ || addr > address(0x03000000000000000000000000000000Ff
        ↪ ));
    }
    // 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(rawTx) [StdCheatsSafe]

```

function rawToConvertedEIPTx1559s(RawTx1559[] memory rawTx) internal pure
    ↪ virtual returns (Tx1559[] memory) {
    Tx1559[] memory txs = new Tx1559[](rawTx.length);
    for (uint256 i; i < rawTx.length; i++) {
        txs[i] = rawToConvertedEIPTx1559(rawTx[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 rawToConvertedEIP1559s(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 rawToConvertedEIP1559(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 cdp;      // Auto incremental
    mapping (uint => address) public urns;   // CDPIId => UrnHandler
    mapping (uint => List)   public list;     // CDPIId => Prev & Next CDPIIds (
    ↪ double linked list)
    mapping (uint => address) public owns;    // CDPIId => Owner
    mapping (uint => bytes32) public ilks;    // CDPIId => Ilk

    mapping (address => uint) public first;   // Owner => First CDPIId
    mapping (address => uint) public last;    // Owner => Last CDPIId
    mapping (address => uint) public count;   // Owner => Amount of CDPs

    mapping (
        address => mapping (
            uint => mapping (
                address => uint
            )
        )
    ) public cdpCan;                          // Owner => CDPIId => 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[cdpi], "dst-already-owner");

    // Remove transferred CDP from double linked list of origin user and
    // → pointers
    if (list[cdpi].prev != 0) {
        list[list[cdpi].prev].next = list[cdpi].next; // Set the next
        // → pointer of the prev cdp (if exists) to the next of the
        // → transferred one
    }
    if (list[cdpi].next != 0) { // If wasn't the
        // → last one
        list[list[cdpi].next].prev = list[cdpi].prev; // Set the prev
        // → pointer of the next cdp to the prev of the transferred one
    } else { // If was the
        // → last one
        last[owns[cdpi]] = list[cdpi].prev; // Update last
        // → pointer of the owner
    }
    if (first[owns[cdpi]] == cdp) { // If was the
        // → first one
        first[owns[cdpi]] = list[cdpi].next; // Update first
        // → pointer of the owner
    }
    count[owns[cdpi]] = sub(count[owns[cdpi]], 1);

    // Transfer ownership
    owns[cdpi] = dst;

    // Add transferred CDP to double linked list of destiny user and
    // → pointers
    list[cdpi].prev = last[dst];
    list[cdpi].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 needed 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-mismatch");
    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-mismatch");
    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-mismatch");
    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-mismatch");
    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);
}
```



12.6.4 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 delivery address
    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 exchnages
    // 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)));
}
```

13.2 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
    ↪ xEeeeeEeeeEeEeeEeEeeEEEEeeeeEeeeeeeeeEEeE;

    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(PsmlLike(psm).gemJoin(), slice);

// Calculate amount of DAI to Join (as ERC20 WAD value)
uint256 daiToJoin = _divup(owe, RAY);

PsmlLike(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 delivery address
    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(
        L2TokenGateway.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:

[illegible]

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 during 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 -- unused
) 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;
}

```


14.5.5 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 receive 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(
        i0VM_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 receive 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, "LiGovernanceRelay/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 receive 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 initiating new xchain messages if bridge is closed
    require(isOpen == 1, "L2DAITokenBridge/closed");

    Mintable(l2Token).burn(msg.sender, _amount);

    bytes memory message = abi.encodeWithSelector(
        i0VM_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 receive 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(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(wards[_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(wards[_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 retrieve 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);
    DSAAuthAbstract(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.ilnToChainlogFormat(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% =>
    ↪ 10000000001243680656318820312)
    @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  $(1 + \text{pct} / 10,000) * \text{WAD}$ 
  @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( 1(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("upchop()")); 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 amount 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( 1(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% =>
↪ 10000000001243680656318820312)
    @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.ilc, co.startingPriceFactor);

    // Set the amount of time before an auction resets.
    setAuctionTimeBeforeReset(co.ilc, co.auctionDuration);

    // Set the allowed auction drop percentage before reset
    setAuctionPermittedDrop(co.ilc, co.permittedDrop);

    // Set the ilc min collateralization ratio
    setIlcLiquidationRatio(co.ilc, 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.ilc, co.kprFlatReward);

    // Set the percentage of liquidation as keeper award
    setKeeperIncentivePercent(co.ilc, co.kprPctReward);

    // Update ilc spot value in Vat
    updateCollateralPrice(co.ilc);
}

```

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-ilc-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. 0(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. 0(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. 0(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. 0(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. 0(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. 0(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. 0(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. 0(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. 0(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. 0(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=
    ↪ W02052xM3AzUCL7o7Pfkow&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=
    ↪ W02052xM3AzUCL7o7Pfkow&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=W02052xM3AzUCL7o7Pfkow&s=19
    ↳ W02052xM3AzUCL7o7Pfkow&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);
}
```