OpenST Protocol v0.9.3 Proposed sequence diagrams for stake and mint - Benjamin Bollen, last edit April 8, 2018 OST:ERC20 Branded Token Gate:SK OpenSTValue:SK Foundation:Worker Branded Token:ERC20 Staker:Address CoreUC:SK : approve(gate, amount) : stakeRequest(amount, tokenHolder) : transferFrom(staker, gate, amount) «OST(amount)» emit StakeRequested(staker, amount, tokenHolder alt [accept] : acceptRequest(staker, amount, hashLock)
: stake(uuid, amount, hashLock, tokenHolder) : transferFrom(processor, OpenSTValue, amount+bounty against policy (KYC/AML) «OST(amount+bounty)» store StakingIntentHash HTLC(processor, amount+bounty) emit StakingIntentDeclared(nonce, unlockHeight, StakingIntentHash) HTLC(staker, amount) : rejectRequest(staker, amount) transfer(staker, amount) «OST(amount)» Placeholder for OpenST Mosaic opt : revertStakeRequest(staker, amount) transfer(staker, amount) ${\rm *OST(amount)*}$ loop new_block(blockHeight, stateRoot) : report(UC, blockHeight, stateRoot) $: \ \, \underline{\text{commit}(\text{blockHeight, stateRoot})}$ emit CommittedStateRoot(UC, blockHeight, stateRoot) new_block(blockHeight, stateRoot) : report(VC, blockHeight, stateRoot : commit(blockHeight, stateRoot) $emit\ CommittedStateRoot(VC,\ blockHeight,\ stateRoot)$ $: {\it claimStakingIntentHash}({\it StakingIntentHash}, {\it merkleProof}[], {\it committedBlockHeight})$ getStateRoot(blockHeight) stateRoot @ blockHeight validate proof store valid StakingIntentHash $emit\ Validated Staking Intent Hash (Staking Intent Hash) \\$ $: ConfirmStakingIntent(StakingIntentHash,\ uuid,\ hashLock,\ amountST,\ amountUT,\ stakerProcessor,\ stakingProcessorNonce,\ unlockHeight)$ assert valid pre-image data $\begin{array}{c} : \ getLatestHeight() \\ \hline & \ latestHeight \end{array}$ assert grace period before unlockHeight store mint with expirationHeight $emit\ StakingIntentConfirmed(StakingIntentHash,\ nonce,\ unlockHeight,\ expirationHeight)$ With both value chain and util-[Proceed]

ity chain configured correctly for StakingIntentHash, Staking Processor can proceed by revealing the unlock secret, or revert by awaiting the unlock height