# #

Software Upgrade

## #

Concepts

### #

Plan

TheUpgrade module defines aPlan type in which a live upgrade is scheduled to occur. APlan can be scheduled at a specific block height or time, but not both. APlan is created once a (frozen) release candidate along with an appropriate upgradeHandler (see below) is agreed upon, where theName of aPlan corresponds to a specificHandler . Typically, aPlan is created through a governance proposal process, where if voted upon and passed, will be scheduled. TheInfo of aPlan may contain various metadata about the upgrade, typically application specific upgrade info to be included on-chain such as a git commit that validators could automatically upgrade to.

### #

Sidecar Process

If an operator running the application binary also runs a sidecar process to assist in the automatic download and upgrade of the binary,Info allows this process to be frictionless. That is, the upgrade module implements the specification specified bycosmovisor upgradeable binary specificationopen in new window , andcosmovisor can Site selection is used to fully automate the upgrade process for node operators. By filling theInfo field with the necessary information, the binary can be downloaded automatically. Refer tohereopen in new window .

type Planstruct { Namestring Time time. Time Heightint64 Infostring UpgradedClientState* types. Any}

### #

Handler

TheUpgrade module facilitates upgrading from major version X to major version Y. To accomplish this, node operators must first upgrade their current binary to a new binary that has a correspondingHandler for the new version Y. It is assumed that this version has fully been tested and approved by the community at large. ThisHandler defines what state migrations need to occur before the new binary Y can successfully run the chain. Naturally, thisHandler is application specific and not defined on a per-module basis. Registering aHandler is done viaKeeper#SetUpgradeHandler in the application.

type UpgradeHandlerfunc ( Context, Plan) During the execution of eachBeginBlock , theUpgrade module checks whether there is aplan that should be executed (the block height or time when theBeginBlock is scheduled to run). If it exists, execute the correspondingprocessor . If the plan is expected to be executed but the corresponding processor is not registered, or the binary upgrade is too early, the node will gracefully panic and exit.

### #

StoreLoader

TheUpgrade module also facilitates store migrations as part of the upgrade. TheStoreLoader sets the migrations that need to occur before the new binary can successfully run the chain. ThisStoreLoader is also application specific and not defined on a per-module basis. Registering thisStoreLoader is done viaapp#SetStoreLoader in the application.

func UpgradeStoreLoader( upgradeHeightint64 , storeUpgrades* store. StoreUpgrades) baseapp. StoreLoader If there's a planned upgrade and the upgrade height is reached, the old binary writesUpgradeInfo to the disk before panic'ing.

type UpgradeInfostruct { Namestring Heightint64 } This information is critical to ensure theStoreUpgrades happens smoothly at correct height and expected upgrade. It eliminates the chances for the new binary to executeStoreUpgrades multiple times everytime on restart. Also if there are multiple upgrades planned on same height, theName will ensure theseStoreUpgrades takes place only in planned upgrade handler.

Currently in the upgrade process, the state transition methods involved mainly support the following three types:Renamed ,Deleted , andAdded

### #

Renamed

The user can specify to migrate all data under oldKey (prefix) to storage under newKey (prefix) during the upgrade process.

#

Deleted

Users can delete all data under the specified key (prefix) during the upgrade process.

#

Added

Users can apply for a new storage area with the specified key as the prefix during the upgrade process.

#

Proposal

Typically, aPlan is proposed and submitted through governance via aSoftwareUpgradeProposal . This proposal prescribes to the standard governance process. If the proposal passes, thePlan , which targets a specificHandler , is persisted and scheduled. The upgrade can be delayed or hastened by updating thePlan.Time in a new proposal.

type SoftwareUpgradeProposalstruct { Titlestring Descriptionstring Plan Plan}

#

Governance Process

When a upgrade proposal is accepted, the upgrade process is devided into two steps.

#

Stop network consensus

After the software upgrade proposal is accepted, the system will perform pre-upgrade preparations at the designated height of the BeginBlock stage, including downloading the upgrade plan and suspending network consensus.

#

Download the upgrade plan

In order to be able to upgrade the software smoothly, the information required for the upgrade must be recorded before stopping the network consensus:plan name ,upgrade height .

-Plan name : When the network restarts, it needs to be routed to the correspondingUpgradeHandler andUpgradeStoreLoader according to the plan name. -Upgrade height : When the network restarts, check whether the network upgrade plan is required.

#

Suspend network consensus

After the software upgrade proposal is accepted, the system will gracefully suspend the network consensus at the specified height of theBeginBlock stage.

#

Restart the new software

The user replaces the software with the specified version and restarts the network. The system will detect whether theprocessor specified by theplan name is included. If it is included, the system first executes theHandler program, and then starts the network consensus. If not, the system report an error and exit.

#

Cancelling Upgrade Proposals

The upgrade proposal can be cancelled. There is a proposal type ofCancel Software Upgrade (CancelSoftwareUpgrade) . When this type of proposal is voted through, the currently ongoing upgrade plan will be removed. Of course, this needs to be voted and executed before the upgrade plan is executed.

If the current upgrade plan has been executed, but there are problems with the upgrade plan, then the proposal ofCancel Software Upgrade is invalid (because the network has stopped consensus). At this time, there is another solution to make up for this mistake, which is to use the--unsafe-skip-upgrades parameter to skip the specified upgrade height when restarting the network (not really skip the height, but jump via software upgradeHandler ). Of course, this requires that 2/3 of the validators participating in the consensus perform the same operation, otherwise the network consensus cannot be reached.

## #

Upgrade process

### #

Submit an upgrade proposal

The first step in the implementation of the software upgrade process is to initiate a software upgrade proposal by the governance module. The proposal details the upgrade height or time. For details, see the aboveConcept . An example of the command line to initiate a proposal is as follows:

iris tx gov submit-proposal software-upgrade< plan-name> \ --deposit < deposit> \ --upgrade-time< upgrade-time> \ --title < title> \ --upgrade-info< upgrade-info> \ --description < description> \ --from = < from> \ --chain-id= < chain-id> \ --fees = 0 .3iris\ -b block

### #

Deposit and vote for the proposal

The execution process of the software upgrade proposal is basically the same as that of other ordinary proposals. Both validators and delegators are required to comment on the proposal. For specific information, please refer togovernance module . An example of the command line to deposit the proposal is as follows:

iris tx gov deposit< proposal-id> < deposit> --from = < from> --chain-id= < chain-id> --fees = 0 .3iris-b block-y Once the deposit amount reaches the minimum deposit amount, the proposal will enter the voting period, and the validator or delegator needs to vote on the proposal. An example of the command line to initiate a vote is as follows:

iris tx gov vote< proposal-id> < option> --from = < from> --chain-id= < chain-id> --fees = 0 .3iris-b block-y When the software upgrade proposal is passed, the upgrade module will create an upgrade plan to stop all nodes from the network consensus at a specified height or time, and wait for the new software to restart the network.

### #

Restart the network

When the upgrade proposal is passed and the network reaches the specified upgrade block height or time, all nodes will stop producing blocks. and the user needs to download the source code and compile the new software according to the new version information specified in the first stepSubmit Upgrade Proposal , refer toInstall . After the new software is installed, restart the node with the new version, and the node will execute the upgrade logic corresponding to the plan name. Once the voting power of the entire network exceeds 2/3 and restarts the network using the new version, the blockchain network will re-reach a new consensus and continue to produce new blocks.