Guide for Initializing genesis blocks

Starting from a clean Computer

With either native or a VM Linux (I use Ubuntu) Install the PTS and BTC clients, then and download the block chains and get them in sync. Once you are synced, check the number of PTS currently in existence and the current block number, so go to the debug window in the client and type "getinfo".

Honouring the Social Consensus Software License

The Social Consensus Software license states that "All chains shall be derived from ProtoShares, with 10% of the ultimate money supply of that chain". To honour the consensus, just allocate 10% of the total value to PTS holders this can be done by scaling 1:10.

Parsing the PTS Block Chain

To import the balances from PTS you must create a list of all addresses and their corresponding balances. Parse the PTS chain into a csv file use the tool below to do the job for you.

This:- https://github.com/vertoe/pts-unspent

Does the job pretty well. the creator vertoe also placed it on the net where it updates daily, so you need not parse the chain, just go here :- http://q30.protoshar.es/pts/pts_unspent.json and copy all the addresses and theirbalances.

Next open the file in software like Microfoft Excel. Once open, click File, choose the Save As option, and as the Save as type: select the CSV (Comma delimited) (*.csv) option.

Honouring and allocating AngelShares

AngelShares are donations made in PTS and BTC by contributors. If your DAC is sponsored by I3 then the funds are AGS and you must allocate 10% of equity to them.

AGS does not have BTC style block chain as a result you will need to view the log of all donations made to the relevant addresses here:-

http://www1.agsexplorer.com/masterbook/btc

http://www1.agsexplorer.com/masterbook/pts
This gives you a complete list of all addresses of AGS holders and how much each of them has.

Save all of the gathered data in a single file named "genesisbalances.txt" with addresses and their balances seperated by “,” and each set in a new line.

If your code base is a fork of PTS or any other code base you need to add the following to your rpcdump.cpp:-

```
string convertAddress(const char address[], char newVersionByte){
        std::vector<unsigned char> v;
        DecodeBase58Check(address,v);
        v[0]=newVersionByte;
        string result = EncodeBase58Check(v);
        return result;
```

And change

```
Value importprivkey(const Array& params, bool fHelp)
{
    if (fHelp || params.size() < 1 || params.size() > 3)
        throw runtime_error(
            "importprivkey <protosharesprivkey> [label] [rescan=true]\n"
            "Adds a private key (as returned by dumpprivkey) to your wallet.");

    string strSecret = params[0].get_str();
    string strLabel = "";
```

To

```
Value importprivkey(const Array& params, bool fHelp)
{
    if (fHelp || params.size() < 1 || params.size() > 2)
        throw runtime_error(
            "importprivkey <NoirSharesPrivkey> [label]\n"
            "Adds a private key (as returned by dumpprivkey) to your wallet.");
    string strSecret = params[0].get_str();
    printf("before %s",strSecret.c_str());
    strSecret = convertAddress(strSecret.c_str(),149);
    printf("after %s",strSecret.c_str());
    string strLabel = "";
```

Pay close attention to the number in bold, you must change it to your PUBKEY_ADDRESS + 128.

Initializing the Balances in a Block

In your main.cpp **add the following** :-

```
std::map<std::string,int64> getGenesisBalances(){
	std::map<std::string,int64> genesisBalances;
	ifstream myfile ("genesisbalances.txt");
	char * pEnd;
	std::string line;
	if (myfile.is_open()){
		while ( myfile.good() ){
			getline (myfile,line);
			std::vector<std::string> strs;
			boost::split(strs, line, boost::is_any_of(","));
			genesisBalances[strs[0]]=strtoll(strs[1].c_str(),&pEnd,10);
		}
		myfile.close();
	}
	return genesisBalances;
}
```

 and then in **CBlockTemplate* CreateNewBlock add :-**

```
 if(pindexBest->nHeight+1==1){
	//Block 1 - add balances for beta testers, protoshares
	std::map<std::string,int64> genesisBalances= getGenesisBalances();
	std::map<std::string,int64>::iterator balit;
	int i=1;
	int64 total=0;
	txNew.vout.resize(genesisBalances.size()+1);
	for(balit=genesisBalances.begin(); balit!=genesisBalances.end(); ++balit){
		//printf("gb:%s,%llu",balit->first.c_str(),balit->second);
		CBitcoinAddress address(balit->first);
		txNew.vout[i].scriptPubKey.SetDestination( address.Get() );
		txNew.vout[i].nValue = balit->second;
		total=total+balit->second;
		i++;
	}
	printf("Total ...%llu\n",total);}
```

Scaling Balances for your Total Equity

DACs sponsored by I3 must allocate a further 10% to AGS holders. There will only ever be 2 million PTS. Even if your DAC has stake, it must maintain the 10% allocations.

So, for sponsored DACs it means you must allocate 10% + 10% of your equity when you initialize the chain. So, once you have the total number of PTS and have calculated the required equity cap, you then allocate an equal number to AGS.

Example

Total PTS---> 1.5 million
Total equity = PTS x 10 = 15 million
AGS = 10% Total equity = 1.5 million
AGS + PTS = 3 million

Example2

Total PTS---> 1.5 million
Total equity = PTS / 10 = 150k
AGS = 10% Total equity = 15k
AGS + PTS = 30k

Your remaining equity is free to distribute as you see fit. Once you have done all the above, you are ready to import the balances into your genesis. If you wish to scale up the numbers or scale down, you need only multiply or divide the balances by whatever factor so long as you maintain the percentages.