Proposed DAC Design

DRAFT

EDNA DAC has the following primary functions which are to be supported by the DAC contract, user interfaces and supporting software integration with (IPFS and EDNA Telegram bot code)

1. **Membership Management**
2. **General Proposal Management**
3. **Custodial Proposal Management**
4. **Referendum Management**
5. **Service Management**
6. **Research Proposal Management**
7. **Custodial Management (Elections)**

This document will take up each of these in turn and address the requirements of the software which will support the DAC activities.

Membership Management

Membership in the DAC requires 1 EDNA token be sent to the ednazmembers EOS account. Members are also required to have a telegram account to allow for communication between members and to allow the membership to vote on proposals, research proposals, services offered to the DAC and custodial representation. The **membership table** has the following fields:

```
struct members{
  uint64_t              member_id;
  account_name          account;
  uint8_t               status;
  asset                 stake;
  string                telegram_user;
  uint32_t              proposal_count;
  uint32_t              vote_count;
  uint32_t              completed_service_count;
  uint32_t              completed_service_value;
  uint32_t              research_opt_in_count;
  uint32_t              research_value_earned;
  asset                 total_funds_earned;
  asset                 member_balance;
  crypt_string          IPFS_traits_data;
  crypt_string          IPFS_gen_data;
  uint32_t              joined_date;
};
```

The member Id field is determined by the EOS next available key function which generates a unique number for this field. It is used in other tables to associate a given member with other tables (proposals, votes and so on).

The account field stored the EOS account which the member wishes to use to manage the membership in EDNA with, it must have at least a 1 EDNA token balance

The member status field is used to track the current status of the member, and enables/disables other functionality based on member status. allowed statuses as as follows:

```
const uint8_t   MEM_MEMBER = 1;
const uint8_t   MEM_PAID_KIT = 2;
const uint8_t   MEM_KIT_SHIPPED = 3;
const uint8_t   MEM_KIT_IN_LAB = 4;
const uint8_t   MEM_DNA_ON_CHAIN = 5;
const uint8_t   MEM_LIFETIME = 6;
const uint8_t   MEM_SUSPENDED = 7;
const uint8_t   MEM_BANNED = 8;
const_uint8_t   MEM_ABANDONED = 9;
```

The stake field holds the assets the member has staked to the DAC. At the time of go-live, this will require 1 EDNA as a minimum. No benefits are established at this time for staking more than 1 EDNA.

The telegram user is a required for membership field and used heavily by the DAC model to communicate with members, and to collect voting activity from members. It is the members responsibility to keep the this field updated and correct so that communication can occur, and votes properly tallied. EDNA User Interface will allow for updating this field.

The proposal count field stores a record of the total number of times the member has proposed a **general** item of business to the DAC.

The vote count field stores a record of the number of times a member votes on an issue before this DAC. This count is incremented regardless of the type of vote being cast (**general**, **service**, **research**, **custodial**, **election** or **referendum**).

The completed service count field tracks the number of times the member has contributed a completed service to the DAC. Services are initiated though a service proposal that is then approved or rejected by the DAC process, and completed by the member. Contributing, but not originating members also are credited with a completed service count. (see completed service value below).

The completed service value field tracks contributions to the DAC both paid and unpaid. Originators of service proposals that solicit or receive contributions from other members are responsible for allocating the service value equitably among the contributors of the service.

The research opt in count field tracks the number of times the member elected to opt-in on a research proposal, this count is incremented regardless of the members ultimate actual participation on the research.

The research value earned field tracks the total rewards paid to the member which were earned by research participation.

The total funds earned field stores a record of the overall earnings of a member regardless of where those earnings originated (service, research or other).

The member_balance field stores funds earned by the member. Members may transfer from this field to the accounts(s) of their choosing, and at their own discretion.

The IPFS traits data field stores an encrypted string, which points to a likewise encrypted file on the IPFS file system. This file contains information about the members physical and psychological attributes, medical history, family history, life experiences and behaviors. This file is not accessible without the cryptography key stored in the members EOS wallet. Members have the ability to produce a sub-set of this data for packaging, re-encryption and subsequent delivery to either a research project approved by the DAC and opted-in on by the member, or to the DACs' internal research program, which likewise requires member authorization and opt-in.

The IPFS gen data field stores an encrypted string, which points to a likewise encrypted file on the IPFS file system. This target file contains a series of pointers to other IPFS files scattered across the world. These other files contain a genetic variations from the GRCh38 Standard Reference Human Genome as maintained and published by the National Center for Biotechnology Information (NCBI).

<BACKGROUND>

The contents of the smaller files (there my be as many as 3 million per member) are as illustrated below, the files are named with a unique name generated post sequencing during the delta_grch38_gpu processing done at EDNA LABS.

File Name: 2bz2qkw4HLsm0rRNvlBKJoSgF4j6NHN9MqyKN1ym

File Contents: [chromosome] [molecular location] [variation]

The 2 examples below in green show different methods of storing genetic variations on the system. Example #1 shows a single base pair where it varies from the CRCh38. Example #2 shows a string of 10 base-pairs where they are likewise variant, but also contiguous. The method shown in Example #2 is **NOT ALLOWED** to be employed in locations used by CODIS and other forensic databases to determine genetic identity or familiar relationships. Variations in these locations must be stored as individual variations in separate files, so as to prevent reconstruction and identification of an individual stored within the EDNA system.

EXAMPLES

PAH gene
(phenylalanine hydroxylase)

Cytogenetic Location: 12q23.2, which is the long (q) arm of chromosome 12 at position 23.2
Molecular Location: base pairs 102,836,885 to 102,958,410 on chromosome 12 (Homo sapiens Annotation Release 109, GRCh38.p12) (NCBI)

EXAMPLE #1 [12][102,900,885][A-G]

EXAMPLE #2 [12][102,840,800-102,840,808][A-G,A-G,C-T,T-C,A-G,G-A,A-G,A-G]

The IPFS gen data field stores the hash of the encrypted “main” file.

Key (main) file. The has of this file is stored int the member table in the field

File Name: uvLUwMeEQKx1UL5qbF0vLZfZqDPLO84V4jqYKZgX

File Contents:
x1xyf3ccXPKq4HLnfB0wIFn5aZVkzCUneiPYnbfh
3qjD9slortYGkUZ3olZ4CvK74KlpyaBye8TrkB4x
omLNnoDe9sdp37EfVaYOQYSShqAK99TCe2WvefFA
uvLUwMeEQKx1UL5qbF0vLZfZqDPLO84V4jqYKZgX
<up to 3,000,000 other file names follow>

<END OF BACKGROUND>

The joined date field stores the date the member joined EDNA DAC

Membership Management Functions

void add(*account, status, stake, telegram_user,* joined_date);

void update(*member_id, account, telegram_user, stake*);

void set_status(*member_id, status*);

void add_proposal_count(*member_id*);

void add_vote_count(*member_id*);

void add_completed_service_count(*member_id*);

void add_completed_service_value(*member_id*);

void add_research_opt_in_count(member_id);

void add_research_value_earned(member_id);

void add_total_funds_earned(*asset.amount*);

void add_member_balance(*account_name from, asset _add*);

void transfer_member_balance(*account_name send_to, asset amount*);

void store_traits_data (*string IPFS_hash*);

void store_gen_data(*string IPFS_hash*);

General Proposal Management

General proposals are **created** by the membership at large using the EDNA proposal bot in the EDNA Telegram room. The have a configurable lifespan before being removed from the general proposal EDNA bot to make room for the next general proposal produced by the membership.

Their “time to live” as a live issue and be voted on by the membership is set within the configs table of the EDNA DAC contract

```
uint32_t      mem_vote_time2live = (60 * 60 * 24 * 7);
```

General proposals are not automatically adopted by the DAC even when passed. They must achieve a certain percentage of voter participation before they are brought before the custodial body for debate, and ratification or defeat. This required percentage of participation is likewise configurable in the configs table of the contract. The setting shown below determines that 40% of the total membership must vote on a general proposal before it is escalated to the elected custodial body

```
uint64_t      proposal_escalation = 40;
```

General proposals are not to be used to propose offering doing work or service for the DAC (there is a separate type of proposal for that), nor should they be confused with proposals from outside researchers wanting to purchase genetic information from EDNA. To that end, there are 5 types of proposals within the contract:

```
const uint8_t   GENERAL_PROPOSAL = 1;
const uint8_t   CUSTODIAL_MATTER = 2;
const uint8_t   MEMBER_REFERENDUM = 3;
const uint8_t   SERVICE_PROPOSAL = 4;
const uint8_t   RESEARCH_PROPOSAL = 5;
```

This section of the document is concerned with type = 1 above. A **general proposal** can have one of the following statuses within the system at any given

time. Only those marked in **bold** apply to **general proposals**. When a general proposal is first created it is set to the GEN_NEW status. It's outcome can be one of two: it may end up in an GEN_UNSUPPORTED status if the "time to live" (explained above) expires without the membership participation in the vote on the proposal reaching the escalation level configured above (in that example 40%). Proposals in unsupported status are not considered by the custodial body, unless they determine the proposal should be picked up and addressed by the custodial body.

The other possible outcome for a **general proposal** is that the required escalation level of membership participation is reached before the "time to live" is expired. It does not matter the vote outcome (for or against), what matters here is that the community has decided by voting that this is a matter of importance and therefore must be addressed by the elected custodial body. The status on the proposal is then flipped to GEN_ESCALATED and the matter is placed in the custodial queue to be addressed. Items having this status are required to be addressed by the custodial body, and there are time limits imposed on them to insure they are taken up.

```
const uint8_t   GEN_NEW = 0;
const uint8_t   GEN_UNSUPPORTED = 1;
const uint8_t   GEN_ESCALATED = 2;
const uint8_t   CUSTO_DEFEATED = 3;
const uint8_t   CUSTO_ PASSED = 4;
const uint8_t   CUSTO_STALLED_1 = 5;
const uint8_t   CUSTO_STALLED_2 = 6;
const uint8_t   CUSTO_STALLED_3 = 7;
const uint8_t   REF_CREATED = 8;
const uint8_t   REF_DEFEATED = 9;
const uint8_t   REF_PASSED = 10;
```

Custodial Proposal Management

Once a general proposal is is converted to a custodial matter, a new time to live value is brought to bear on the proposal. This value is likewise stored in the configs table and is configurable.

```
uint32_t      custodian_vote_time2live = (60 * 60 * 24 * 2);
```

Currently, this value is configured for 48 hours. The custodial value therefor has 2 days with which to address the escalated proposal by discussion, debate and ultimately vote. One of 3 outcomes are possible once a proposal has reached this stage. It may be passed or defeated (CUSTO_ PASSED / CUSTO_DEFEATED) by the custodial body, or it may move into a state of CUSTO_STALLED_1

in order for the custodial body (of 12 elected individuals) to pass or defeat a proposal a configurable majority must vote for or against it. This majority is also stored in the configs table:

```
uint8_t       custodial_majority = 9;
```

so, in the matters before the custodial body a vote of 7 to 5 is insufficient to pass or defeat a proposal. A majority of 9 must be reached for the vote to be considered actionable. In such a case, the proposal is moved to a status of CUSTO_STALLED_1 where the time to live clock is reset, and debate resumes on the matter. If the matter is not resolved by required majority vote within the time to live, the clock again resets and the status of the matter is set to CUSTO_STALLED_2 whereby the process repeats one final time. Should the proposal not be decided by the required majority after 3 tries by the custodial body, the proposal is altered a final time to a **referendum**. And escalated back to the membership at large to be terminally handled.

All custodial discussion, debate and voting on active proposals is to be conducted in the EDNA DAC custodial telegram group. This group is open to all EDNA DAC members, but is read-only for all but elected custodians.

Should an elected custodian create a proposal it will initiate with a status of GEN_ESCALATED so that the custodial body may move to address it.

Referendum Management

Referendums are considered to be the most critical vote within the EDNA DAC system. They are passed by the majority requirement as configured in the contract config table, and would require that number of participating votes (percentage of total possible) in order to pass, or be defeated. Referendums have no time to live associated with them, and may continue to be discussed and debated on till they are resolved.

```
    uint64_t      referendum_passage = 51;
```

Proposals of the type General Custodial and Referendum live on the same table within the EDNA DAC contract. They are held on this table until suck time as they expire, are defeated or approved, and move to EDNA Archives Status. The EDNA Archive table is a record of document stored on the IPFS system and held in the archives table by storing their IPFS file hash.

Active (non-archived) proposals are stored on the proposals table presented here:

```
   struct proposals {

    uint64_t          prop_id;
     uint8_t          prop_status;
     string           prop_title;
     string           prop_ipfs_text;
     uint64_t         prop_sponsor;
     uint64_t         prop_next_action_date;
};
```

The *prop id* field is determined by the EOS next available key function which generates a unique number for this field. It is used in other tables to associate a given proposals with other tables as needed.

The *prop status* field is used to manage the progress of the proposal though the system. From it can also be derived what type the proposal is currently, and can be used to present the data in various displays on the EDNA DAX user interface.

The *prop title* field stores a name to refer to the proposal by in forms, reports and other displays.

The *prop ipfs text* field is used to store on the table the ipfs filename hash. This value is used to retrieve the text of the proposal from the ipfs file system for display.

The *prop sponsor* field stores the member_id value from the members table of the DAC member that initially proposed the proposal.

The *prop next action date* field is used to move the proposal along thought the various statuses when those state changes are brought about by elapsed time (such as 48 hours for custodial action). The contract will make use of a deferred action function which will trigger every hour and seek out next action dates that are in the past compared to system time. It will then take appropriate actions on the adjustment of the status of the proposal as well as resetting the next action date where that step applies.

Proposal Functions

void add_proposal(*prop_id, prop_status , prop_title, prop_ipfs_text , prop_sponsor, prop_next_action_date*);

void set_proposal_status(*prop_id, prop_status, prop_next_action_date*);

void_archive_proposal(*prop_id*);

Service Management

Service Management is a slightly different function than proposal management in that it typically involves the allocation and expenditure of DAC funds. Any DAC member can propose a Service to the DAC, but it is not routed first to the overall membership, does not route through the custodial over-watch process designed to force a decision, and can not ultimately result in a referendum placed before the DAC Body. The statuses of a service offering is are a bit more intuitive than proposal statuses and are presented here:

```
const uint8_t   SERVE_NEW = 1;
const uint8_t   SERVE_DENIED = 2;
const uint8_t   SERVE_APPROVED = 3;
const uint8_t   SERVE_APPROVED_WITH_MOD = 4;
const uint8_t   SERVE_MOD_ACCEPTED = 5;
const uint8_t   SERVE_MOD_REJECTED = 6;
```

```
const uint8_t   SERVE_IN_PROGRESS = 7;
const uint8_t   SERVE_COMPLETE = 8;
const uint8_t   SERVE_SETTLED = 9;
const uint8_t   SERVE_UNDELIVERED = 10;
```

Service offerings are routed directly to the Custodial body voting queue, where a majority of 7 is sufficient to approve them. The Custodial body may elect to approve a service offer with modifications to the original offer as received and in that case the offer is routed back to the originator so that they may accept or reject the modifications.

The table must track additional information for a service offering, so it is a bit more involved than a proposal. Services that are presented to the Custodians with a zero cost are assumed donated services to the DAC.

The general membership may “rate a service” and the service will maintain both a count of times rated, as well as the average rating for that service.

```
  struct services {
      uint64_t              service_id;
      uint64_t              member_id;
      uint8_t               service_status;
      string                ipfs_service_descr;
      asset                 service_cost;
      uint64_t              start_date;
      uint64_t              projected_end_date;
      uint64_t              actual_end_date;
      uint8_t               average_rating;
      uint32_t              times_rated;
}
```

```
void add
void set_status
void modify
void pay
void rate (member_id, rating)
```

Research Proposal Management

The research proposal management process is to allow outside genetic researchers to propose used for the data owned and stored by the members of EDNA DAC. The process is designed to allow both custodians and members serving on an ad-hock approval committees to vet the research, funding entities, purposes and likely outcomes as far as possible before approving the research proposal and making it available to the general membership for their determination of participation when those members meet any criteria set my the offer. The approval committee will also negotiate any payments and ownership rights stemming from proposal participation to ultimate findings, products or services produced from the research.

Research proposals do not shift from group to group as other proposals but live and die with the committees formed to study and approve or deny them. Members my self-nominate their participation in the a research committee for any given proposal or number of proposals. Committees once formed, must produce a service proposal in order to earn compensation for service to the DAC.

```
    struct research {
      uint64_t                research_id;
      uint64_t                edna_sponsor_id;
      string                  project_title;
      uint8_t                 research_status;
      string                  company_entity;
      string                  contact;
      string                  email;
      string                  telephone;
      string                  website;
      uint64_t                number_genomes_sought;
      uint64_t                number_of_opt_ins;
      uint8_t                 genome_type;
      string                  ipfs_hash_traits_sought;
      string                  ipfs_hash_funding_source_disclosure;
      string                  ipfs_hash_research_purpose;
      string                  ipfs_hash_publications_list;
      string                  ipfs_hash_notes_comments;
      asset                   proposed_payment;
      asset                   surety_bond_posted;
      uint8_t                 profit_sharing_offered;
      uint8_t                 agree_terms_of_service;
      uint8_t                 agree_edna_intended_use_policy;
      uint8_t                 agree_edna_constitution;
      uint8_t                 agree_eos_constitution;
}

struct research_committee {
      uint64_t                committee_id;
      uint64_t                research_id;
      uint64_t                chairman_id;
      uint64_t                member1_id;
      uint64_t                member2_id;
      uint64_t                member3_id;
      uint64_t                member4_id;
      uint64_t                member5_id;
      uint64_t                member6_id;
      uint64_t                member7_id;
      uint64_t                member8_id;
      uint64_t                member9_id;
      uint64_t                member10_id;
      uint64_t                member11_id;
}

struct research_member_x_ref {
      uint64_t                research_id;
 uint64_t               member_id;
}


const uint8_t     RESEARCH_NEW = 1;
const uint8_t     RESEARCH_UNDER_REVIEW = 2;
const uint8_t     RESEARCH_DENIED = 3;
const uint8_t     RESEARCH_APPROVED = 4;
const uint8_t     RESEARCH_ACCEPTING_OPT_IN = 5;
const uint8_t     RESEARCH_IN_PROGRESS = 6;
const uint8_t     RESEARCH_COMPLETE = 7;
const uint8_t     RESEARCH_PAID = 8;
```

void join_committee(member_id);

void update_committee_member(member_id, member_id);

The committee chairman shall be responsible for updating the status of the research progress as this is a manual process and can not be driven by time or other events. The chairman will default to the DAC member who initially sponsors the research proposal, and who is recorded as the sponsor of the project in the research table, however this can be updated and re-assigned at a later time as the project progresses.

The sponsor / chairman shall cooperate to insure the documentation of the system is complete and that the DAC is in possession of signed agreements from the responsible parties as indicated by checking the “agree” boxes in the user interface corresponding to the table fields. The sponsor / chairman shall insure the requirements of documentation are stored on the IPFS files system and may assist the researchers in storing that documentation.

The one exception for maintaining this table is when enough members opt-in on the research project that the number equals the genomes sought value, at that time the status is changed to RESEARCH_IN_PROGRESS and further opt-in’s are not allowed.

Custodial Management (Elections)

```
// election statuses
      const uint8_t   ELECT_OFF = 0;
      const uint8_t   ELECT_NOMINATING = 1;
      const uint8_t   ELECT_VOTING = 2;

// custodian statuses
      const uint8_t   CUSTO_NOMINATED = 0;
      const uint8_t   CUSTO_DECLINED = 1;
      const uint8_t   CUSTO_RUNNING = 2;
      const uint8_t   CUSTO_DEFEATED = 3;
      const uint8_t   CUSTO_SITTING = 4;                    // auto-nominated
      const uint8_t   CUSTO_REMOVED = 5;                 // can not be nominated
      const uint8_t   CUSTO_RETIRED = 6;

// custodian config table
      uint64_t      nominations_time2live = (60 * 60 * 24 * 3);
      uint64_t      elections_time2live = (60 * 60 * 24 * 2);
      uint8_t       custodian_count = 12;
      uint8_t       custodial_majority = 9;    // 9 of 12 must approve or defeat
      uint32_t      custodian_time2live = (60 * 60 * 24 * 90); // time 2 elections
      uint8_t       percent_incumbent_replace = 60;
      uint8_t       impeachment_majority_precent = 65;
      uint8_t       removal_majority_percent = 51;
      uint64_t      next_cust_election_due; // date-time of next elections
      asset         dac_funds_main;
      asset         dac_funds_approved_spend;
```

A word about the custodians of the EDNA DAC:

Custodians are EDNA DAC members who have agreed to put extra work and time into EDNA. They will have diverse knowledge, skills-sets and opinions on how the

DAC should operate. Primarily, they should be individuals of high character, and should be motivated by a desire to serve the DAC and promote the EDNA Constitution, and to hold the well-being those things above their own self-interests.

The membership body should realize that they are ultimately responsible for who they elect to serve, and even the actions of those they empower. The EDNA system has been designed with the highest degree of transparency as a primary design consideration, and has been done so in an effort to insure the member body has great ease in monitoring the actions of those they elect.

No custodial business is to be conducted outside the EDNA Custodial Telegram Room. This room and its' history shall serve as the source of truth and system of record for the custodial operation of EDNA. While phone calls, emails, video meetings and other methods of communication are from time to time needed for expediency, they are only permitted to the degree that their contents may be (and are in fact) replicated within the telegram room record. Evidence of non-compliance with this policy of complete transparency is grounds for immediate and irrevocable removal from service to the DAC.

Process

Any member may nominate any other member (or themselves) for custodial service. The nominated member may accept or decline the nomination.

When accepting the nomination, the candidate should have a written biography, a face picture and ideally a 10-minute or less campaign video describing why they want a custodial seat and anything else they feel would get their message across. The system provides hosting and access to that video (1 per candidate), and bio.

Nominations are currently configured to run for 5 days then close and elections will begin. 2 days would be allowed for voting, and then the matter decided.

Elections are to be held every 90 days.

A maximum of 60% of the incumbent seats may be replaced in a single election, should the sitting custodians opt to retain their seats – obviously if the sitting custodians opt to not run or retire this number can be exceeded, but in general efforts should be made to retain some knowledge of how operations have been run in the past – especially when those operations are deemed successful by the membership at large (see impeachment section below for member recourse when the the membership believes the custodial body needs a "clean house").

NOTE: all of the above time-frames are parameterized – and can be adjusted (nomination window, election length of time, election frequency and sitting replacement percentage).

The graphical interface used by the membership for voting will show the following information about each candidate:

1. Candidates **Voting History** on all DAC matters, General Custodial, Service and Referendum.

2. Candidates **Proposal History** on all matters as above.

3. Candidates **Election History** showing previous custodial service and election participation.

4. Candidates **Service History** showing all service proposals initiated by the candidate, approved or not, compensated or not.

5. Candidates **Biography, photo & Contact Information**.

6. Candidates **Promotional Video**

Impeachment Proceedings: The general membership may at anytime call for impeachment of one or more sitting custodians. Currently this is configured at 65% of the existing membership must vote “yes” to impeach a sitting custodian(s). When this vote results in the required majority. The election system shall be initiated to nominate, and elect a new special custodian to conduct the proceedings. Once elected, the new custodian shall collect and present evidence and testimony acting as both representation for the accuser and accused. Sitting custodial members not under impeachment may likewise collect and present evidence and testimony, as well as question any member with knowledge of the events leading to the accusation(s). Once the evidence and testimony have been placed into the record (using the custodial telegram), the special custodian shall initiate a referendum level vote for the membership at large. A 51% membership majority shall be sufficient to remove the custodian(s) in question following the proceedings.

CONCLUSION

This ends the first draft of the proposed DAC contract. Efforts have been made to construct it as flexibility as possible to allow the DAC overall ability to respond to changes in its future needs. It is submitted with full certainly that it will be modified from this initial effort.

The full table structure, and action (code functions) should not be considered complete – some modifications will be required to fully execute the intention of the design. This document will be maintained as development proceeds and is likely subject to multiple additions as the code is brought up to full ability of serving the designs intent.

Spending procedures of DAC controlled funds has been largely omitted from this document – the initial tools of the DAC will not be constructed as to dictate how that business is conducted. Coding efforts will be executed with maximum flexibility in mind – so that the DAC, once established, can determine the procedure which best serves.