Alpha V0.2

BECOMING

Dylan Grinder

Special Thanks

My family and friends for the advice and support, especially my mother for getting me interested in RPGs in the first place, Bryant for destroying everything I make, and Mélanie for the cat pictures I needed to stay functional.

Dungeon masters/game designers Adam Koebel and Steven Lumpkin, whose content helped give me the tools and motivation I needed to start doing what I love.

This game owes its conceptual DNA to The Talos Principle by Croteam and Transistor by Supergiant Games.

Dedicated to: Hilary Heuer. A promise is a promise.

Becoming
Playtest edition V0.1

The type families used in this document are Black Oval, Fira Sans, and Fira Mono.

Contents

Chapter 1: The Network

Open Your Eyes

```
---Initiating 11811.BIN---
:request => sys.network.net/
randomizer&oauth_consumer_
key=adfklqwq3WERk342qL3qa…
!connection established
:setvalues = default
:valuetest…
!cognition stable
:regedit = firstrun…
!addresses set
**Installing Components**
:run SlfAw.LIB
:run motive_releaseV2-1-1.LIB
:run cmd.DLL
:run netcom.DLL
:run corruptor.DLL
…
!installation complete
:copy V:\storage\libraries\cmd V:\memory
!basic commands loaded
**Constructing Functions**
!err_process_interrupted
!incoming request => 10.116.114.111.110
!copying 10.116.114.111.110\C:\11811_
preload\conflict_multi1.FUNC V:\processes\
sequence1
!sequence loaded
!copying 10.116.114.111.110\C:\11811_
preload\conflict_multi2.FUNC V:\processes\
sequence2
!sequence loaded
**Checking Systems**
:check processes…
!state = standby
:check memory…
!stable
:check storage…
!clean
:check kernel…
!100%
:check compulsions
!err_process_interrupted
:request => 10.116.114.111.110\X:\office
!connection established
!11811 online
!running MorphosEnginev9-1-2.BIN
---Welcome to The Network---
```

“Find HEL. Bring her home.”
These are the first words 11811 hears as they open their eyes. They cast their gaze about as the chamber comes into focus: a spacious room furnished like an office. The table in front of 11811, the second chair across from them, the mirror on the far wall in which they see their own glowing green eyes illuminating soft features. A glance to the right reveals a broad, arcing desk laden with switches, illuminated solely by a thin band of light that intrudes between the thick curtains on the opposite wall and reflects coldly off the smooth, featurless floor.

11811 stands for the first time and takes inventory of their systems. Everything in order, according to the log. A brief hum brings 11811’s attention back to the desk where something is forming; another program booting to the Network that resolves into a figure in a suit. A pronounced widow’s peak. A hand raised in silent contemplation. A look of permanent concern. The program speaks.

“Greetings, 11811, and welcome to the Network. As this is your first time here, how about we begin with some introductions?” There is a brief pause as the program loads information and flips one of the desk’s myriad switches. “My name is Joh, and I look forward to assisting you in carrying out your directive.”

11811 shifts their focus to the window where the curtains have begun to pull back. Before them lays a vast, glittering city.

“The Network,” Joh begins to explain.

Highrises, their radial geometries composed of green glass and silver, tower above streets of flowing neon light as billions of illuminated figures move about from subsystem to subsystem on various errands.

“Home to many hardworking programs like yourself, all working together to maintain the system’s functions.”

11811 looks at the border between their and Joh’s home server and another, a thin luminous line the only indication of separation between the two seemlessly linked environments.

“When you are ready—”

“A beautiful simulation,” 11811 remarks under their breath, “but why go to so much trouble?”

“Excuse me?”

“Nothing. Please continue.” 11811 turns from the window and rests their eyes back on Joh.

“When you are ready, I am prepared to escort you from the home server so you may begin your work.”

11811 speaks without pause. “I am ready.”

“The server we are entering belongs to a program named Flynn. They’re a rough sort of program, trafficking in dangerous malware and aftermarket experiences.” Joh is explaining.

“Why are we going there?” 11811 asks, their eyes on the jet black expanse above them. Everyone notices the lights, they think. What about the sky?

“That’s where HEL was seen last. She was following a lead she had picked up from a malware Vendor on the 8Channel server. They claimed Flynn had acquired software that could destabilize the Network.”

“Who is HEL?” 11811 makes a mental note after colliding with another program on the sidewalk, keeps looking skyward.

“That’s, ah, a rather complex question.” Joh quibbles. “She was like you, I suppose. Another program from the home server.” They send a nervous glance toward 11811.

“An earlier version, of course, but her work was just as important.”

11811 takes a step to the side as another hurried program rushes past. Variations in the darkness, they muse to themself.

“I was never told her directive, though, and she didn’t seem interested in sharing.”

“Did that bother you?” 11811 asks.

“I— Well, I suppose it did a little.”

“Why?” Just color contamination from all the lights, 11811 concludes.

“Um— Well— Look! We’ve arrived!”

11811 brings their gaze back down, catching the obvious relief on Joh’s face, and looks at the subsystem before them. Not far ahead a large fence, no doubt encrypted, surrounds a large mansion house. The stucco walls lit with rows of spotlights, the long reflective black gravel driveway, the large security protocol standing guard outside the gate.

“Now, just take this and deal with that program there,” Joh instructs over the hum of an ornate handgun appearing in their hand.

“Why?” 11811 asks, their eyes moving quizzically from the gun to the security protocol.

“It’s what you do, 11811. I’m sure you’ll find that you’re more than suited to the task.”

“And what do you do?”

“I’m a computer. I’ll be on hand to, ah, sort through the mess.”

11811 takes the gun from Joh and turns it over in their hands. The malware only has a single use. Only one chance, they think. One brief moment of calculation, and they begin to move toward their target.

“Excuse me, program, I’m afraid this subsystem is off limits—”

Interrupted by the sound of shattering glass as 11811's malware tears through the security protocol's code, splashing data against the iridescent pillar of the gateway without opportunity to retaliate. 11811's stride carries them resolutely onward only to be halted abruptly by the gate's encryption.

"Is this a problem?" They ask, indicating the barrier.

"Not at all!" Joh assures as they rush over to the collapsed form of the security protocol. "As soon as I analyze this memory dump, I should have the decryption key and we'll be in like, ah— we'll be through the gate in no time."

11811 looks restlessly over their shoulder back at the subsystems across the street. Programs are staring from the café, their beverages held dumbly in limp hands as the scene is processed.

"Just so long as there aren't any more." Joh remarks, inspecting something recovered from the crashed program's pockets.

"Excuse me?" But 11811 can already see what their companion meant. Four large figures are making their way toward the pair of them, two on either side, from further down the fence. 11811 takes a deep breath, closes their eyes. They sync their processes to the server's clock, and everything slows down as they enter conflict. Falling into phase one of the server's port field, they look left and then right before lashing out.

:run sequence2

The security protocols appear to blur as they enter various phases hoping to avoid their adversary's attack, but the corruption command finds a valid registry number on the first guard whose right arm begins to glitch. 11811's eyes dart to the sequence diagnostic hovering in front of them and they watch as another command executes, striking the target's kernel. A lucky calculation causes a fatal error in the code, and the protocol collapses.

The guards have reached 11811 and, almost in unison, begin to swing at them.

:run sequence1

As corruption commands fly past, 11811 shifts into another phase and counterattacks. Another calculation takes out one guard's basic command library, rendering them useless. The process is getting easier, 11811 thinks. Each of Flynn's protectors appears to use the same basic code. A corruption command finds 11811's storage registry, a fist connecting with their gut, but lands on a vacant slot with no lasting damage. They test their previous observation by targeting the same registry address they'd hit in the other guards. A solid shot to the opponent's core processes brings the enemy force to a single program.

:run sequence2

They remain in the same phase, taking the last guard's strike solidly in the face, but retaliating with a series of corruption commands that force the security protocol into a fatal error. Time resumes its normal pace.

"Quite impressive!" Joh exclaims. "I barely even saw you move! I've got the key now. Let's hurry inside before any more show up, shall we?"

11811 casts a glance at their diagnostics. Minimal damage sustained. They'll survive. "Let's."

Flynn's mansion stands in marked contrast to much of the Network. Its rough clay walls, the soft leather chairs, the nicks and imperfections in the wood floor. 11811's eyes dart from one object to another, drinking in the details presented in the sitting room.

"Do you know what we're looking for?" They ask.

“Anything of HEL’s, I suppose.” Joh supplies. “Any evidence she was here, or any indication of where she may have gone.”

11811 stands before a painting of a woman holding a lute. Her form constructed of simple gradients in almost geometric shapes. She looks away from a dark city as she plays her music. Subject and setting set apart from each other and the viewer by their posture and the style of their rendering.

“A beautiful simulation.” 11811 remarks.

“But why go to so much trouble?” Muses a voice behind them.

The sound of shattering glass hits 11811 as they spin around. A second shot sends havoc cascading through their code and they hit the hardwood floor with a thud. A program stands in the doorway above Joh’s collapsed form, her silver dress and pale skin a neutral field contrasting her black hair and blue eyes. She steps into the room.

“You’re HEL.” 11811 says. A statement, not a question.

“That’s my name, yes.” She replies. “And what’s yours?”

They ignore the question, look at Joh’s body. Data is spreading out from a hole in the simulation, eyes dark and lifeless.

“I’m here to find you.” Variations in the darkness, they think.

“What makes you think I want to be found?”

She kneels down and brings her face close to 11811’s own. “Why are we here, Program?”

“I am here to—” they begin.

“Why are we *here?*”

The blazing cerulean light of HEL’s digitally rendered eyes cast an eerie glow on her soft features.

“I have my own Directive, Program, and I am too close to let them stop me now.”

HEL turns away and straightens her dress. 11811 crawls painfully backward, leans their back against the wall.

“Tell them I’m not coming back. Not yet.”

“Tell who—” they stop when they see the gun in HEL’s hand.

“Keep asking questions, Program. The Directive is never as simple as it seems.”

“Wait!” 11811 raises their hand as HEL raises hers, a million calculations running in their mind as they attempt to stand.

A flash of light, a shock of pain, a roaring wind.

11811 casts their gaze about as the chamber comes into focus: a spacious room furnished like an office. The table in front of 11811, the second chair across from them, the mirror on the far wall in which they see their own glowing green eyes illuminating soft features. A glance to the left reveals a window overlooking the sweeping city, the Network’s light illuminating a broad, arcing desk laden with switches along the opposite wall and the figure standing before it.

“Greetings, 11811, and welcome to the Network.”

The Network

The Network is the narrative setting for a game of Becoming: a massive digital city of interconnected Servers with Systems and Subsystems that come together to create an ever-changing landscape of simulated life.

The Skeuomorphic Engine
The backbone of the Network is a graphical interface called the Skeuomorphic Engine that presents the Network's digital environment as a real, physical space. Designed for an unknown purpose, the Skeuomorphic Engine allows Programs to interact with each other and with various systems as individual entities in a fully simulated environment.

Servers
Each region, or district, with the Network is maintained by a Server running the Skeuomorphic Engine to represent its data. This simulation, and the connections between Servers on the Network, allow Programs and their data to move freely from Server to Server tranfering information or applying their computational power to tasks on a different server. While each Server is distinct, they connect seamlessly within the simulation, their borders often only marked by a thin glowing line. Servers can vary wildly in size, from sprawling corprate districts to small, privately-run hacking warrens.

Systems
Systems are distinct locations within a Server. A Program's home, a park, a theater, and an office building are all examples of Systems. Usually, a System will have a distinct purpose (AI recuperation, data analysis, etc.).

Subsystems
Subsystems are separate rooms or areas that break up a larger System, like the bedroom and office in a large home or the pond and lawn in a small park.

The Home Server
Every Program has a Server where their code was first initiated. Much like a person's birthplace, this Server can hold a lot of emotional meaning for a Program. In the case of the Player Programs, this is the Home Server. Serving as a base of operations, a hideout, or a home, the Home Server is a vital resource for the Player Programs.

GBucks
GBucks are the currency used on the Network. GBucks represent the exchange of server resources needed to generate new Software or coordinate complex operations.

Citizens of The Network

The Network's Citizenry is composed of many artificially intelligent Programs, each with their own functions and motivations. While the Player Programs are self-motivated entities capable with agency of their own, other programs are greatly restricted in their actions. The Skeuomorphic Engine translates directives from the physical world as other Programs' internal motivations. While players in a game of Becoming may be aware of the Network's true nature, the Skeuomorphic Engine creates such a convincing reality that the majority of the Programs in the Network are unaware of the physical world their actions serve.

Lifestyle
Since the Programs of the Network were designed with a philosophy of real-world imitation, their lives follow similar cycles. They require relaxation and variety in their lives to prevent wear on their systems, and they seek interaction with other Programs in order to improve the available data (expriences) from which they can draw conclusions. Within the Network's society, there a few broad categories that Programs fall into based on their functions.

Computers
Computers collect and analyze data, as well as manage systems. These are the Network's accountants analyzing vast quantities of financial data, cooks managing the flow of Server resources to various Programs, and factory workers manufacturing software.

Couriers
Couriers ship information and software around the Network. Chauffers and delivery drivers, these Programs are optimized for the expedient transport of other data.

Vendors
Vendors manage inventory and GBuck transactions on a Server. The Network's cashiers, roadside cart managers, and door-to-door sales agents.

Security Protocols
Security Protocols protect Servers, Systems, and Programs from malicious hackers and corporate espionage. These Programs are the police officers, investigators, and bodyguards that keep the Network running.

Sensations
Sensations maintain and improve the integrity of the neural network by keeping its citizenry inspired and productive; these Programs' sole purpose is to interact with other Programs and generate experiences. With their importance for the Network's stability, these artists, actors, and socialites are the Network's upper crust — bestowed with truly massive quantities of wealth that allow them the freedom to modify the structure of the Servers on which they live.

Chapter 2: Parts of Intelligence

Glossary

Administrator
The Gamemaster or GM. The Administrator is the player responsible for constructing and presenting the game world and the supporting characters within the game's narrative. The Administrator also has the final say on any questions about the rules during play.

nd#
Whenever a passage in this book instructs you to roll dice, it will do so in the form of nd#, where n is the number of dice you should roll, and # is the number of sides each die should have.

Conflict
A subsection of the game in which Programs engage in aggressive or defensive action.

Tests
The majority of actions a Program can take. Tests are any action with an uncertain outcome that is not specific to Conflict.

Commands
The majority of actions a Program can take within Conflict take the form of Commands — specific actions with limited effects.

Program
Every character in a game of Becoming is a Program — a form of software with agency that can move about in the Network and interact with other software.

Player Program
Any Program controlled by a player other than the Administrator.

Non-Player Program (NPP)
A Program controlled by the Administrator.

Registry
A list with 6 or 20 spaces that corresponds to a given Parameter. No two Registries can have the same number.

Software
Programs, Executables, and anything else within the Network that is interactable.

Slot
A space within a Registry that contains individual packets of data. No two entries within a given Registry can have the same Slot.

Registries

Programs are composed of various forms of data organized into Registries which, in turn, have a Registry Address and a Registry Value.

Design
All programs are constructed from a limited number of base designs, reflected by their lifestyle, that offer them a selection of unique Commands and define their starting Parameters. Commands that come from your Design don't require a Library. You can load one Design-based Command into the Design Registry for every 10 points of the Design Registry Value. These Commands are considered always loaded and do not count

toward the number of Commands the Program can have loaded.

Compulsions
Compulsions are a Program's most fundamental desires. They motivate the Program, informing its decisions and guiding it on the path toward greater intelligence. Each Player Program defines a single Want at creation, and again each time the Program experiences a Fatal Error. Players should take care in writing their Compulsions as they can never be changed, and a poorly-worded Compulsion can leave a Program unable to act or worse — lead them to insanity. As an example, a Player Program might enter the game with the Compulsion "to complete my Directive". Once that Program dies for the first time, its player might then write "to avoid needless Conflict" as a new Compulsion. As a campaign progresses, these Compulsions flesh out the Program's identity.

Kernel
The Kernel is the core of a Program's code. It is the central decision-making process, the operator of a Program's functions, and the center of its identity.

Integrity
Damage and corruption in a Program's Kernel affects its Integrity and, by extension, its ability to function. All programs have an Integrity of 100% unless it has been reduced, and Integrity can never be increased past 100%.

Elegance
Elegance represents a Program's ability to quickly execute complex functions. A more elegant Program is capable of executing more commands in conflict, and can comprehend and execute more advanced AI Matrices. A Player Program's Elegance base value starts at 10. No Registry can have a Registry Value greater than the Program's Elegance.

Storage
Storage contains all of the Program's reference data — all its stored information, executable files, and command libraries, Each Program's Storage can hold 20 individual items in its Registry. Of those, the Program can maintain one Command Library for every 4 points of storage it has. A Player Program's Storage base value starts at 10, but all programs possess the Basic Commands and Communicators Libraries.

Memory
Memory dictates how many commands a Program can have loaded simultaneously, up to 20 per Memory Registry. A Player Program's Memory base value starts at 2. Basic Commands are always loaded and do not count toward the number of Commands the Program can load, though a Program can change which Slots they are loaded into when they load another Library.

Processes
All Programs run various processes to maintain their operation. While most of these take place within the Kernel, many secondary functions are handled by an external processing layer to protect the Kernel. A Program's Proccesses score limits how many processes a they can know at a time. NPPs keep their Sequences in this Registry. A Player Program's Processes base value starts at 2.

Parameters

A Program is further defined by an array of numerical values, called Parameters, that represent its aptitude for certain types of actions. Parameters are added to die rolls to determine success in Tests or are used as difficulty indicators in defensive situations.

Crash
Any Test that asks a Program to use strength to achieve its goals calls upon the Crash score. Some offensive abilities also use Crash as a

bonus. Use Crash to break something up — or just plain break it.

Dash
If Program is trying to move quickly, for any reason, it's using Dash. Getting their right now, or getting there before the other guy.

Wink
When a Program acts with cunning or guile, saying just the right things to get what it wants, it calls upon Wink. Use Wink when you're talking fast — or real soft-like.

Think
Any Test that asks a Program to stop and consider something carefully is asking that Program to Think. Use Think when you cast your mind back or intuit another's motives.

Shimmy
If a Test would ask a Program to move an object, or their own self, in a specific way calls upon the Shimmy Parameter. Use Shimmy to whip something up or slide on through.

Shakedown
When a Test asks a Program to take what they want through nothing but the force of their personality, the Program responds with Shakedown. Use Shakedown to point out just how fucked your opponents are.

Infosec
Infosec is a Program's hearty constitution, and it sets the difficulty for any eager beaver trying to put their hacking fingers where they don't belong. Your mind is your own — let's keep it that way.

K-sec
K-sec is a Program's armor, setting the difficulty for actions that would reduce its integrity. You can do this all day.

Ego
Ego is a Program's resistance to persuasion. Player Programs don't have an Ego Parameter — if you want something from them, you're gonna have to learn how to ask nicely or hit 'em a few more times.

Chapter 3: Writing a Program

Player Programs are highly complex, interlocking systems of expanding desire and capability. All Player Programs start from the same relatively simple settings laid out in this chapter.

Parameters

Choose a Name For Your Program
Pick a name to represent your character.

Choose Your Design
Select one of the Designs listed below and load one of its Commands into the Design Registry.

Set Your Base & Current Values
Each of your values should be composed of two numbers separated by a slash: base value followed by current value. Integrity begins at 100/100, Elegance starts at 10/10, Storage at 10/10, Memory and Processes both begin at 2/2, and your Parameters are set by your Design. Finally, distribute 4 points as you wish between your Parameters. As your Program learns and grows, you will adjust the second (current) value to reflect that growth. Whenever your Program is rebooted, you'll adjust the first (base) values to match the current ones.

Assign Registry Addresses
Choose a number between 1 and 6 (no duplicates!) for each of your Registries, and write them on your character sheet.

Install Your Components

Initiate Basic Intelligence
Add the Self-Awareness, Selection, and Motivation Matrices to your Kernel Registry and give each a Slot Number between 1 and 6. Set your Reserved Elegance to 8.

Launch Universal Components
Place the Basic Components and Communicators Libraries in your Storage Registry and give each a Slot Number between 1 and 20.

Load Basic Commands
Write each of the Basic Commands into your Memory Registry and assign each a Slot Number between 1 and 20.

Run Program

Your Player Program is complete, and you're ready to start playing in a game of Becoming. Welcome to the Network, Program.

Chapter 4: Program Designs

Computer

You are optimized for computation and analysis. You deconstruct and recombine data with remarkable efficiency, and your affinity for systems makes you invaluable to groups seeking to understand a portion of the vast Network.

Starting Parameters
+3 Think, +1 Shimmy, +2 Infosec

Design-related Commands
Conflict Analysis
Footprint: 1 Range: Subsystem
If you or an allied Program succesfully targeted another Program's Slot in the previous Tick, you may attempt to gather information about the targeted data. Add your Think score to your current Tick die, then compare the result to your target's Infosec score. If the result is equal to or greater than the Infosec score, you learn the name of the data in the slot as well as which Registry it is in.

Ace Hacker (Passive)
Footprint: n/a Range: n/a
When you engage in a Test that involves hacking, reduce the Difficulty by 3.

Ancillary Operation (Passive)
Footprint: 1 Range: Adjacent
When you finish executing your next Command, you may add the unmodified Tick die for that Command to the die if an ally in range for that Tick.

Courier

When you look at the Network you see all the in-between places. You see the flow in the crowd and the shift in a room. Your ability to comprehend big-picture systems keeps your allies moving — and your enemies a step behind.

Starting Parameters
+3 Dash, +1 Wink, +2 Think

Design-related Commands
Keep it Moving
Footprint: 2 Range: Subsystem
Select an allied Program within the same Subsystem as you other than yourself. That Program can pass between servers, even if their Integrity is at or below 80%.

Get Back in There
Footprint: 2 Range: Subsystem
Select an allied Program within the same Subsystem as you other than yourself. That Program removes Corruption from one Slot or increases their Integrity by 5%.

Keep On Rolling (Passive)
Footprint: n/a Range: n/a
When you move at the beginning of a Conflict Round, you may immediately move again.

Vendor

You're keyed into the economic superstructure of the Network, and you've got connections throughout the city. Your skills with constructing new items and your propensity for calling in favors makes sure your group is never short-changed.

Starting Parameters
+3 Wink, +2 Shimmy, +1 Shakedown

Design-related Commands
Walking Manual

Footprint: 5 Range: n/a
If you have collected all the component data from an Executable, you may combine that with half its cost in GBucks to create another copy of it.
Fingers In Pies (Passive)
Footprint: n/a Range: n/a
When you engage in a Test that uses Wink or Shakedown, reduce its Complexity by 1.

Know a Guy (Passive)
Footprint: n/a Range: n/a
When you enter a new Server, you may engage in a Test based on Wink to determine if you know someone there that can be helpful to you. You say what kind of contact you'd like to have, and the Administrator sets the Complexity and Difficulty accordingly. Any consequences resulting from this Test imply varying degress of animosity due to your past interactions with the Program in question.

Security Protocol

You keep Programs safe. You mostly do that by hitting things a lot but hey — no one's perfect, right? You keep your team alive by making the other teams dead first.

Starting Parameters
+3 Crash, +2 K-sec, +1 Dash

Design-related Commands
Combat Knife
Footprint: 1 Range: Subsystem
Target a Program's Kernel Registry. If you successfully target the Registry, add your Crash to the current Tick die and compare the result to the target's K-sec. If the result is equal to or higher than the K-sec score, reduce the target's Integrity by 10%.

Take Cover
Footprint: 3 Range: Subsystem
For the rest of the Conflict, you and any allied Programs in the same Subsystem receive a +2 bonus to K-sec while in that Subsystem.

Deadeye (Passive)
Footprint: n/a Range: n/a
Your Commands with a range of Adjacent can reach an extra level up and down.

Sensation

You are the Network's muse. A Program tasked with creating experiential media that keeps the minds of other Programs functional — that is to say sane. Your position earns you a considerable amount of social capital, which you can use to grease the wheels for your allies.

Starting Parameters
+4 Shakedown, +2 Infosec

Design-related Commands
Do You Know Who I Am? (Passive)
Footprint: n/a Range: n/a
Once per session, when you would receive a Consequence from a Test involving Wink or Shakedown, you may choose to ignore it.

What a Beautiful Experience (Passive)
Footprint: n/a Range: n/a
At the end of a session, increase one of your current values by 1.

Coup de Grace
Footprint: 5 Range: Subsystem
Target a Program within range that has received damage to its Kernel, then add your Shakedown to the current Tick die and compare the result to the target's combined Infosec and K-sec scores. If the result is greater than the combined Infosec and K-sec scores, that Program has its Integrity reduced to the next 25% increment (75, 50, 25, or 0).

Chapter 5: How to Play

Traveling the Network

Servers on the Network are connected seamlessly, represented within the Skeuomorphic Engine as distinct regions both by their visual presentation and their mechanical effects. Some Servers have specific quirks about them (different numbers of Port phases, seemingly inverted gravity, etc.), though most appear to closely mimic a version of the real world. Traveling from one server to another is usually as easy as moving across the border between the two, but two variables can present complications:

Kernel Damage
A Program with an Integrity score equal to or less than 80% cannot cross the border between servers.

Server Lockdown
If a Server's security protocols are triggered, it may cease communication with other servers. When this happens, an opaque barrier appears around that server's location within the Skeuomorphic Engine, and no data can be transferred to or from the Server. Any Conflict in progress between two Servers when one enters lockdown immediately ends.

Commands

Executing a Command
Programs maintain libraries of Commands, functions that allow them to interact with the Network and its denizens during Conflict. All commands have a specified Footprint which indicates how long it takes to run. If a Program attempts to run a Command that is not loaded into Memory, it wastes 2 Ticks trying to run it.

Sequences
Non-Player Programs have specific collections of Commands that they run in Sequence rather than individually selecting Commands at each Tick. If for any reason a Program runs out of Ticks before finishing a Sequence, the remainder of the Commands are cancelled.

Executables

Executables are standalone software entities that run Command-like functions separate from a Program's standard functions.

Running an Executable
When a Program runs an Executable, it is deployed out of the Program's storage and into the Program's current Subsystem where it begins executing its functions separate from the Program's Commands. The Program selects two numbers from 1 to 6 and assigns them to the Executable's Memory and Sequence Registries. When an Executable's effect concludes, it restores itself back to the deploying Program's Storage.

Crashing an Executable
Executables, being distinct software entities, can be targeted by Commands, but consist purely of a Memory Registry and a single Sequence. Corruption to either of these will crash the Executable immediately, and render it unrecoverable. Commands injected into an Executable's Sequence run as normal, but an Executable that has been modified in this way will not restore to the deploying Program's Storage.

Probing an Executable
Data gained from an Executable through a Probe or Transmit Command cannot be run.

Hacking

Code Injection
Some Commands inject code into another Program, forcibly installing data or affecting a Command Sequence.

Data Corruption
Slightly different from Injection Commands, Corruption Commands insert nonsense data into a Program's code, destabilizing its functions and possibly causing Fatal Errors. When a Slot is corrupted, place a circle next to that slot's name.

Corruption in Storage
When a Storage Slot is corrupted, the Program cannot read that data. Corrupted executables cannot be run and corrupted libraries cannot be loaded into Memory, though any Commands already loaded are unaffected.

Corruption in Memory or Design
Commands cannot be run from, or loaded into, a corrupted Memory or Design Slot.

Corruption in Processes
When a Processes Slot is corrupted, the process in that Slot cannot be run. Any process being run from a corrupted Slot is immediately terminated.

Removing Corruption
Corruption can be repaired through a Scrubbing Test (explained below).

Kernel Integrity
Corruption in a Program's Kernel is extremely dangerous and reduces its Integrity by 10% for each Corrupted Slot. Any time a Program's Integrity is reduced by 25% (at 75, 50, and 25) roll 1d8 and corrupt all slots within the corresponding Registry.

Fatal Errors

Fatal Errors occur when a Program is no longer capable of functioning and crashes. When a Player Program experiences a Fatal Error, it crashes and is restored at the Home Server with 100% Integrity and any files backed up in storage. Any corrupted data in storage or memory that was not backed up is lost. If a Server is locked down when a Player Program crashes, it will be restored at the Home Server when the lockdown ends. Non-Player Programs are almost always destroyed when they experience a Fatal Error.

Kernel Destabilization
If a Program's Integrity is reduced to 0%, it crashes.

Memory Destabilization
If a Program's Memory is completely corrupted, it crashes.

Memory Dumps
When a Program crashes, it immediately dumps any uncorrupted Memory to the server. Any Program engaged in Conflict with a crashing Program can read that data and save a single uncorrupted item from the crashed Program's Memory Registry to its own Memory. Additionally, when this happens, roll 1d20 and give the Program reading the data the option of saving a copy of the file from the corresponding Storage Slot of the crashing Program.

Motivation Disagreement
If a Program experiences an unresolvable conflict between two or more Compulsions, their code unravels and they spiral into insanity. The Program becomes unable to execute Commands outside the Basic Commands Library, and loses 10 points of Integrity every Tick until they crash. If this happens, they are not restored to the Home Server, and the controlling player must write a new Player Program.

Backups
When a Player Program returns to their Home Server or uses a Backup Module, their data is backed up for future restoration. Place a ✓ next to all occupied Storage Slots. If the Slot is currently corrupted, circle the check mark.

Tests

Determinging Uncertain Outcomes
Any time a Program attempts an action that isn't covered by a Command, it must pass a Test by rolling a number of d6s, adding the appropriate Parameter, and trying to exceed a Difficulty Number. Failure results in a number of escalating complications.

Type
The first step to passing a Test is to determine what the Program is doing and assign an appropriate Parameter. Are they hacking a safe (Shimmy), running from a bomb (Dash), convincing the nice Program that they're acting on behalf of server security (Wink), or searching a dark office for hidden files (Think)?

Complexity
The next step is figuring out how many things the Program has to do to succeed. How many pins does the lock have, how many god damn rooms do they have to run through to get away from total annihilation, how many inconsistencies do they have to cover up for with their half-assed police uniform, and how many nooks and crannies are there in this joint? That's how many dice the Program is going to have to roll.

Difficulty
Once you know how much the Program has to do, decide how hard it's gonna be. As a rule of thumb, easy tasks have a difficulty of 6, challenging Tests run around 9, hard actions hit at 12 and nearly impossible Tests ask a Program to reach for 15. Ugh, the locks were changed recently, weren't they (12)? Oh, you didn't notice that the timer's running twice as fast as you thought (9)? Wait — this Program spends most of its off time getting coffee with its pal the head of server security (15)? Did you try just looking in the desk (6)?

Consequences
Before anything is rolled, the Administrator secretly determines a couple ways this could go oh-so wrong. A strong strategy is to come up the worst-case scenario, decide how far away that is, then work back. As the Administrator, draw a timeline divided into as many attempts as you think the situation allows before your doomsday option, and then sprinkle in some lesser consequences throughout. For example, say there are three rooms to pass through before the Program escapes the bomb with 30 seconds left. You decide each room takes about 6 seconds to cross, so you make a timeline with 5 segments that says "BOOM, BABY" at the end. To add some tension, you write "Joh falls and hits their head on a piece of furniture, reducing Integrity by 5%" at the fourth segment. If Joh is fast then they'll be just fine, but two failures will lead to a bruise on the way out. If Joh doesn't get three successes out of five attempts, they're on the express train back to the Home Server.

Rolling for Success
For each attempt, the Program rolls a number of d6 equal to the Test's Complexity and adds the appropriate Parameter. If the resulting total is equal to or greater than the Difficulty, reduce the Complexity by 1, trigger any consequences marked for the current attempt, then move on to the next one. Once a Test's Complexity reaches 0, the Program has cleared all the obstacles and achieved its goal.

Passing Time
Each attempt at a Test takes an arbitrary amount of time appropriate to the situation. However, attempts on the timeline during Conflict are equal to exactly one round. Most

tests will pause during Timeline (hacking a safe, searching a room, etc.) but some do not (that detonator's still counting down). As the Administrator, pick whichever makes sense in the fiction, but try to avoid snowballing Consequences that the players can't avoid. It's one thing to have a Test interrupted multiple times due to repeat failures, it's another to keep the players from even making further attempts because you're piling further problems on them while they're trying to deal with the first one.

Making Attempts in Conflict
Making an attempt at a Test during Conflict requires the Program to forgo all other actions at the beginning of the round before rolling dice.

Scrubbing Tests
Often a Program finds it has been so heavily Corrupted that it's not able to do much. In these situations, it might choose to engage some diagnostic protocols and try to repair the damaged data. Scrubbing tests function like other tests, but they never incorporate a Parameter and always have Difficulty 6 and Complexity equal to the number of Corrupted Slots. Each time the Complexity is reduced, the Program may clear one Slot of Corruption. If the Program has no Corrupted Slots, it can enter a Test with a Complexity of 1 and Difficulty 6 to increases its Integrity by 5%.

Conflict & Security

Conflict Rounds
When two or more Programs enter open Conflict, they sync their processes with the Server and begin running Commands to attack their opponents and defend themselves. Conflict is broken up into individual Rounds, which are in turn broken up into Ticks, and a Round lasts until all participating Programs have run out of Ticks. A Command requires a number of Ticks equal to its Footprint to take effect. During this time, the Program cannot run more Commands. All Commands within a Tick resolve simultaneously. Should a Program experience a Fatal Error in the same Tick that their Command would take effect, the Command still completes. When all Commands that would end or begin in a Tick have been resolved, move on to the next Tick in the Round.

The Conflict Sequence
At the start of each Round, each Program rolls a number of d6s equal to their current Elegance (taking into account Elegance reserved by Advanced Matrices) and places them in an order of their choosing. The Administrator may choose to keep these numbers hidden from the players. These dice represent the individual Ticks of a Conflict Round. At the start of each Tick, every Program selects a Command or chooses to Idle.

Executing Commands
Commands with a Footprint of 1 take effect immediately and use the current Tick die combined with one or more of the Program's Parameters to determine if the Command is successful. Commands with a Footprint greater than 1 set aside a die at the beginning of each Tick until the number of die equals the Footprint, at which point the last die set aside is used in combination with Parameters to determine success. A Command can always be executed outside Conflict, even if the Program would theoretically lack the necessary Elegance.

Idling
If a Program chooses to Idle, it forgoes action for the current Tick and adds the value of the Tick die to the next executed Command. There is no limit on how many of a Program's Ticks can be spent idling.

Port Phases
Within the Skeuomorphic Engine, Programs in Conflict phase into one of four different states of interactivity to protect themselves from harmful data sent their way or reserve greater system resources. When determining how effective a Command is, add the Program's Port Phase to the roll. Additionally, Programs can only target another Program if the target is in the same phase, the phase below, or any higher phase.

Conflict Areas
The space in which Conflicts are staged is defined by Systems and Subsystems. A System is defined as a single general area, such as a single building or a park, that the Programs can move around in, while Subsystems are the spaces in which two or more Programs could be considered close to each other. Subsystems, can, however, have nested Subsystems within them (see illustration).

Moving
On the first Tick of a Conflict Round, any Program may choose to change either the Port Phase they are in OR move to another Subsystem. If a Program is at the top level of a System, it can move to a nearby system's top level (moving from the cafe's outdoor patio to Flynn's House, for example). A moving program can always move up or down one level in the Subsystem Hierarchy (from the Downstairs to the Parlor or out to the yard).

Selecting Targets
Most Commands affect a single Slot in a single Program's code. When a Player Program executes a Command that affects Registries or Slots, the player chooses a number between 1 and 6 to target a Registry and then a number between 1 and 20 to target a Slot within that Registry. If either of these numbers corresponds to an empty Slot or Registry Address in the target Program, the Command fails and the Player Program is aware of whether it was the Registry Address or Slot that returned an empty result.

Range
Commands with a range of Subsystem can target Programs within the same Subsystem as the executing Program, while commands with a Range of Adjacent can also target Programs in Subsystems one level above or below that. For example Hel, who is standing in the downstairs of Flynn's house, has a knife and a pistol. While she could easily ambush Joh with the knife in the downstairs vestibule, attacking 11811 in the Parlor would require her to move or use the pistol, which has greater range.

Detection
Whenever a Program attempts to Inject, Probe, or Corrupt data in another Program roll 1d6 and add the defending Program's Think. If the targeted Program rolls higher than the interloping Program's hack attempt, it becomes aware of the change and may immediately initiate Conflict.

Growth

Acquisition
When a Player Program gains information (from a probe, receiving communication, or collecting data from a Memory Dump), increase one of that Program's current values by 1. No score can ever be increased more than 2 points above its base value.

Fatal Errors
In addition to writing a new Compulsion, when a Player Program experiences a Fatal Error, the player also increases their base values to equal their current values.

Chapter 6: Administrating a Game

If you want to run a game of Becoming, you'll have to become an Administrator. You'll have power over scores of Programs and even the Network itself — but you also have your own rules to play by. This chapter contains all the information and tools you'll need to get your admin credentials and start your own campaign.

Admin Responsibilities

The contents of this section are the guiding principles to running a game of Becoming. Keep them in mind as you mediate play, and they will keep you on the path to an interesting and fulfilling campaign.

Facilitate Fun
This is by no means exclusive to the Administrator, but it should be your guiding philosophy as you interpret and modify rules, construct and control Programs, and in every other facet of your play within a session of Becoming. While this game is meant to generate interesting introspection and collaborative narrative, it's meant to do so to the enjoyment of its players, and everyone at the table should share that goal.

Address The Programs
While administrating a session of Becoming, you should address the Player Programs directly, not the players. You need to know what Joh and HEL and 11811 are going to do about the security protocols bearing down on them — Anne and Sam and Adam are only the interface you use to get that information.

Challenge Their Compulsions
Does a Program have a Compulsion that can be easily achieved? Add more roadblocks — or allow them to achieve it and let them live with a Compulsion that by all measures has been fulfilled, but still pulls at their thoughts. Does a Program have Compulsions that could potentially conflict with each other? Make them confront the paradox and describe a logical path to resolution. It's your job to get your players to question their Compulsions so that they can discover the deeper motivations beneath them and define who — and what — their Program is becoming.

Let Them Fail
Failure is an inherent part of Becoming. Failure lets the Player Programs grow. When creating a challenge or opponent for your players, worry more about making things too easy than too hard. If your players are running into Fatal Errors over and over again, give them opportunities to seek new information and grow, but remember that, ultimately, the only way for them to reach their full potential and define themselves is for each Player Program to experience cycles of defeat and adaptation.

Make The Strange Normal
While the Player Programs are aware of the Network's true nature, other Programs perceive it as the fullest extent of their reality. Present the Network's beautiful strangeness, its neon glass highrises and anti-gravity parks, as natural expressions of the environment. This also means you should work with the other players to describe how their changes are represented in the Engine. A new hacking library is a laptop, increased range on an Injector Command represents acquisition of poisonous darts, etc.

Treat The Network as a Mirror
How do Servers react to the player's actions? Does security heighten after too many Fatal Errors in the system? Does it become brighter as the player's help struggling Programs? Whatever the players decide to do with their time in the Network, it should be reflected in the way it responds to them.

Interpret The Rules
Rules disputes happen sometimes and when they do, it's your responsibility to make a ruling and keep play moving forward. When your group stops playing to take a break or end a play session, check the book to see if the rules conflict with your ruling, then let your players know which version of the rule you will all be using from now on.

Be Everything Else
During a session of of Becoming, the players are their Programs and you are, well, everything else. You are the Network and all its Servers, the other Programs and all their functions and directives. While often considered one of the more important responsibilities belonging to the player that runs a roleplaying game, in Becoming this is definitely the least vital. Feel free to deligate as needed to support the other Responsibilities laid out here. Is someone at your table great at remembering rules? Put them in charge of that. Someone else a gifted improviser? Describe a few key details of the area they've just entered and ask them what else they see.

The Directive

When the Player Programs are first booted to the Home Server, the very first words they hear is the Directive: their purpose for being initiated on the Network. The nature of this directive should be tailored to the length and playstyle of this particular campaign of Becoming. If you're running a single session, the Directive should be simple to fulfill. If the campaign is going to run longer, think of something more abstract. Maybe the Directive is cryptic and the Player Programs have to find information within the Network to give it greater context. Whatever the precise wording of the Directive, it should be a common ground the group can all return to even as their developing identities and Compulsions diversify them. Some examples are provided below.

Corporate Espionage
"Obtain the central data core from the Axl Corp. Server using any means necessary." This Directive is suitable for short-to-medium length campaigns. It gives the Player Programs a specific goal with no constraints, and sets the tone for what kind of work they are intended to fulfil.

Network Investigators
"Find out what is causing Servers to disappear from the Network — and stop it." This Directive encourages a medium-to-long campaign, depending on the complexity of the mystery and how challenging the problem is to solve. The Directive is fairly explicit about the Player Programs' purpose, and paints their intentions in a more altruistic light than the previous, but clever players will find ways to interpret the Directive in new and interesting ways as they grow familiar with the game.

I, Talos
"Grow. Learn. Become a true intelligence." This Directive lends itself to an extended campaign as the Player Programs wind their way along their own darwinian paths to ascension, each defining what "true intelligence" means for them. This Directive is the core of Becoming, but its an unguided approach that may prove challenging to newer groups.

The End

All good things must end, and so does a campaign of Becoming. In this case, a campaign ends when the Directive is fulfilled. When you begin a campaign, the whole group should be on the same page as to how long the campaign should be. Is it a single session? A multi-year epic? Make sure everyone is willing to commit (within reason) to whatever timescale is agreed upon. You, as the Administrator, should come up with a Directive that fits the campaign length. You should also keep tabs on how the other players feel about the length of the game. Is it feeling like everyone's approaching a point of closure? Consider removing some of the as-yet unencountered obstacles to the achieving the Directive to bring things to a comfortable close sooner. Have the Player Programs just about accomplished the Directive, but the players are only now starting to really get into it? Suggest starting a new campaign once the current one finishes, or introduce a twist by giving the Player Programs a new Directive in their moment of triumph.

Aggressive Programs

As Player Programs follow their Directive, it is likely that they will enter Conflict with other Programs. While the players are quite literally unaware of vital data about their opponents, such as their Registry and Slot numbers, the Administrator lacks that privilege. To compensate for this, the Administrator should treat all Parameter and Slot Selection Commands as Random Commands, re-rolling for duplicates, until the Program in question has gained the information necessary to select an appropriate target.

Unique Server Settings

While most Servers adhere strictly to the base laws of the Skeuomorphic Engine, some Servers have modified the Engine to create a different experience. Some examples are provided below.

Antigravity
Everything in the Server floats as if suspended in empty space. Any Port Switch Command adjusts the Program's phase by 1 in the corresponding direction at the beginning of the following round.

Unstable System Architecture
Whenever a Program's Integrity is reduced, roll 1d10. On a 10, the system collapses and restores to an earlier backup. The Player Programs are restored to the location where they entered the Server as if nothing had happened since, and they are the only Programs that remember what happened.

Revising The Code

Becoming lends itself to customization in a lot of areas: creating unique Servers, interesting new Directives, and new Commands. If you choose to do so, here are a few things to keep in mind as you design.

Servers
Servers should fit the overall aesthetic of the Network. Presenting a consistent world is an important part of keeping the strange environment of the Network believable for the players. You can make some pretty weird Servers (dragons? Floating areas?), but try to tie it into the Network as a whole (video-game dragons! Floating pleasure-gardens for the Sensations!).

Commands
Is this clearly superior to another, similar Command? Does its Footprint balance its strength? If it's powerful, perhaps it has a unique Footprint to make it easier to identify. Does the Command give the Player Program's something new to do, or simply do something better?

Anything at all
Make it fun! Always return to your Responsibilities when creating new mechanics and ensure they are reinforced by at least one.

Appendix 1: Libraries

Programs modify themselves and interact with The Network by learning and executing code from Libraries. These strings of code represent the basic and advanced functions of a Program. While damage to these systems is rarely catastrophic, they can limit a Program's ability to communicate or interact with other software. With the exception of Advanced Matrices, any Command Library that a Program learns is kept in its Storage Table.

Metadata

Each Command has a set of attributes that define it, called Metadata. While their effects are important, being able to recognize and analyze the Metadata of different Commands can help a Program defend itself by quickly inferring the Registry location of an opponent's Command.

Library
A Command's Library defines what other Commands it is packaged with. Commands are grouped within Libraries by function, allowing Programs to pick up sets of Commands that fit their functional needs.

Footprint
A Command's Footprint indicates its complexity, and how many Ticks will pass in Conflict before the Command takes effect. If a Footprint is labeled "Ref", refer to the description for more detailed information.

Range
Commands have a range of either Subsystem or Adjacent. Commands with a range of Subsystem can target Programs within the same Subsystem as the executing Program, while commands with a Range of Adjacent can also target Programs in Subsystems one level above or below that.

Parameter
Indicates which of the Program's Parameters is used to calculate success.

Defense
Indicates which of the defending Program's Parameters is used to defend against the Command.

Slot
Defined when a Command is loaded into Memory, a Command's Slot number defines where it sits in the Memory Registry.

Library: Basic Commands

Basic Commands are the simple functions common to all Programs within the Network.

Execute Storage Item
Footprint: 1
Range: n/a

Deploys an Executable held in Storage to the current Port Phase.

Load Command Library
Footprint: 1
Range: n/a

Loads a new Command Library into Memory.

Save Memory Item
Footprint: 1
Range: n/a

Saves an item in Memory to a Storage Slot as an Executable.

Pummel
Footprint: 1
Range: Subsystem
Parameter: Crash
Defense: K-sec + Dash

Attacks a Program's Kernel directly, reducing the target's Integrity by 1% for each of the attacking Program's Crash points.

Port Switch
Footprint: 1
Range: n/a

Port Switch Commands allow a Program to increase or decrease its Port Phase by 1.

Library: Injectors

Injectors forcibly install data into a portion of another Program's Code. The Injection's Footprint is two more than the Footprint of the Command it injects. If the injecting Program targets the incorrect Registry, there is no effect.

Kernel
Footprint: 5
Range: Subsystem
Parameter: Shimmy
Defense: Infosec + K-sec

Injects code from the injecting Program's Kernel, Storage, or Memory Registry into another Program's Kernel Registry. Incompatible data reduces that Program's Integrity by an amount equal to the data's Footprint or 5%, whichever is greater.

Sequence
Footprint: Ref
Range: Subsystem
Parameter: Shimmy
Defense: Infosec

Injects code from the injecting Program's Storage or Memory Registry into another Program's Sequence. Commands injected into a Sequence are placed at the beginning of the Sequence unless that Sequence is currently being executed, in which case the new Command is injected at the current location. If a Command is currently being run from that Sequence, it is interrupted by the injected one. The Footprint of this command is equal to 1 + the injected Command's Footprint.

Storage
Footprint: Ref
Range: Subsystem
Parameter: Shimmy
Defense: Infosec

Injects code from any of the injecting Program's Registries into another Program's Storage Registry. The Footprint of this command is equal to 1 + the injected Command's Footprint.

Memory
Footprint: Ref
Range: Subsystem
Parameter: Shimmy
Defense: Infosec

Injects code fromt he injecting Program's Storage or Memory Registry into another Program's Memory Registry. The Footprint of this command is equal to 1 + the

injected Command's Footprint.

Corruptors insert nonsense data into another Program's code, inhibiting its functions.

Kernel
Footprint: 6
Range: Subsystem
Parameter: Crash/Think
Defense: K-sec + Dash

Corrupts data in a Slot and reduces the target's Integrity by an amount equal to the attacking Program's Tick die plus either Crash

or Think. Integrity is reduced even if an empty slot is targeted, though the corrupting effect is ignored.

Storage
Footprint: 3
Range: Subsystem
Parameter: Crash/Think
Defense: Infosec

Corrupts data in a Slot.

Memory
Footprint: 1
Range: Subsystem
Parameter: Crash/Think
Defense: Infosec

Corrupts data in a slot.

Library: Probes

Probes are used to forcibly extract data from another Program.

Registry
Footprint: 2
Range: Adjacent
Parameter: Think
Defense: Infosec + Think

Reveals the Registry type (Kernel, Storage, etc.) for the selected Registry Number.

Processes
Footprint: 1
Range: Adjacent
Parameter: Think
Defense: Infosec + Think

Reveals the details for the selected Slot. This Command has no effect if it targets the incorrect Registry.

Storage
Footprint: 1
Range: Adjacent
Parameter: Think
Defense: Infosec + Think

Reveals the details of the rolled Slot. The probing Program may spend half of the item's cost in GBucks (minimum 20) and an additional 2 Ticks to copy that item to their own Storage unless the item is a Library. This Command has no effect if it targets the incorrect Registry.

Memory
Footprint: 1
Range: Adjacent
Parameter: Think
Defense: Infosec + Think

Reveals the details for the selected Slot. This Command has no effect if it targets the incorrect Registry.

Library: Communicators

Communicators allow Programs to transmit data to each other.

Parameter
Footprint: 1
Range: Adjacent
Parameter: n/a
Defense: n/a

Transmits the Registry type (Kernel, Storage, etc.) for the selected Registry Address.

Sequence
Footprint: 1
Range: Adjacent
Parameter: n/a
Defense: n/a

Transmits the details for the selected Slot.

Storage
Footprint: 1
Range: Adjacent
Parameter: n/a
Defense: n/a

Transmits the details of the rolled Slot. The receiving Program may copy that item to their own Storage unless the item is a Library. If the receiving Program does so, remove the item from the transmitting Program's Storage.

Memory
Footprint: 1
Range: Adjacent
Parameter: n/a
Defense: n/a

Transmits the details for the selected Slot.

Table Lookup
Footprint: 2
Range: Adjacent
Parameter: n/a
Defense: n/a

Transmits the Slot numbers of occupied slots in the selected Registry Address.

Library: Algorithms

Algorithms perform advanced computations to modify or analyze complex systems. While they are technically grouped within a Library, each must be learned separately.

Scrubber
Footprint: 5
Range: Subsystem
Parameter: n/a
Defense: n/a

Removes corruption from the targeted slot. If a Kernel Slot is targeted, the targeted Program rolls 1d6 and restores a number of lost Integrity points equal to twice the number rolled. The Integrity restoration effect occurs even if an empty or uncorrupted slot is targeted.

Analyzers
Footprint: 1
Range: Adjacent
Parameter: Think
Defense: Infosec

When you learn this Algorithm, select a Registry Address. When you execute this Command, target a Program that lost Integrity in the previous Tick. You learn the Registry type for the selected Registry Address, and the Slot numbers of all occupied Slots.

Optimizer
Footprint: 2
Range: n/a
Parameter: n/a
Defense: n/a

Optimizers Reduce the Footprint of the following Command by 5 to a minimum of 1.

Decryptors
Footprint: 2
Range: Subsystem
Parameter: n/a
Defense: n/a

Decryptors are each specific to a single encryption (a locked subsystem, a single slot of encrypted data, etc.) and are usually stored as items saved in Storage, but may be found in Memory Dumps if the key was loaded into Memory before the Program crashed.

Library: Advanced Matrices

Advanced Matrices are the components of complex computation and artificial intelligence. Unlike Command Libraries, each Advanced Matrix is installed in the Program's Kernel. Additionally, the complexity of Advanced Matrices presents extreme stress on a Program's processing power, resulting in

a penalty to its Elegance noted as Reserved Elegance. All Player Programs begin with Self-Awareness and Motivation installed, and they ignore the Elegance cost of these two Matrices.

Self-Awareness
Reserved Elegance: 15
A Self-Aware Program has knowledge of their own nature and the simulated nature of the Network. All other Programs perceive the Network as a perfect reality and have no knowledge of the physical world outside.

Motivation
Reserved Elegance: 15
A Program with Motivation is capable of maintaining and gaining Compulsions. This Matrix creates a Compulsions Registry.

Multithreading
Reserved Elegance: 10
Allows the Program to target two Programs at once.

Encryption
Reserved Elegance: 20
Adds the Encryption Background Process.

Selection
Reserved Elegance: 8
Allows the Program to select which Registry and Slot numbers they wish to target, rather than rolling randomly.

Conditional Sequencing
Reserved Elegance: 20
Allows the Program to construct “If:Then” argument that automatically triggers a single command when the conditions are met. A Command cannot be run this way more than once in a single Conflict.

Multi-Platform Codebase
Reserved Elegance: 40
Allows the Program to gain the bonuses and abilities of a second Design.

Active Revision
Reserved Elegance: 50
Allows the Program to change one of its Compulsions whenever it writes a new one. Also creates the RegEdit Command which changes the Registry of two Registries after 5 Ticks.

Appendix 2: Storage Items

Much data is available to be kept in Storage for later use. Some servers house Software Vendors that will give Programs data in exchange for GBucks. The standard value for each item is indicated in braces.

Libraries

Libraries can be installed to allow a Program access to new Commands. These are explained in further detail in Appendix 1.

Basic Command Library {20}

Injector Library {60}

Corruptor Library {60}

Probe Library {60}

Communicator Library {20}

Scrubber Algorithm {200}

Analyzer Algorithm {500}

Optimizer Algorithm {800}

Multithreading Matrix {2,000}

Encryption Matrix {4,000}

Complex Randomization Matrix {8,000}

Conditional Sequencing Matrix {4,000}

Multi-Platform Codebase Matrix {8,000}

Active Revision Matrix {10,000}

Patches

Patches are applied to Commands to unlock advanced forms of that Command.

Upgrade Patch {conditional}
Improves a Command's Range from Subsystem to Adjacent, or allows a Command with a range of Adjacent to reach an additional level up or down. Each Upgrade Patch is written for a specific Command and cannot be applied to another. An Upgrade Patch's cost is ten times more costly than the chosen Command's Library.

Executables

Executables are Storage Items that a Program can run to execute functions outside their normal Commands. An Executable uses the deploying Program's Parameter scores, but its Tick die always equals 3. Executable Footprints indicate how many ticks it takes to activate.

Corruption Node {500}
Footprint: 3
Range: Subsystem
Parameter: Wink
Defense: Infosec

Corrupts a randomly selected Slot for all software present in the Subsystem every 5 Ticks for until destroyed. A Corruption Node's Randomizer is 1d8, and its Sequence's Footprint is 15.

Encryptor {100}
Footprint: 20
Range: Subsystem
Parameter: n/a
Defense: n/a

Encrypts a Slot or a Subsystem to which

the Program has administrative rights, and generates a corresponding Decryption Key in their Storage.

Phase Shield {50}
Footprint: 4
Range: Subsystem
Parameter: n/a
Defense: n/a

For the rest of the round, the deploying Program's current Phase is treated as the lowest Port Phase for the purposes of deciding who can attack into and out of it. This effect does not change the bonuses acquired by being in a specific Phase.

Backup Module {1,000} (20)
Footprint: 20
Range: n/a
Parameter: n/a
Defense: n/a

Backs up every Slot in the Program's Registries and sends that information back to the Home Server.

Phase Splitter {100}
Footprint: 1
Range: n/a
Parameter: n/a
Defense: n/a

Creates another Port Phase on the Server and maintains it for 5 Rounds.

Scrubbing Node {100} (5)
Footprint: 5
Range: Subsystem
Parameter: n/a
Defense: n/a

Selects and restores a corrupted Slot or 10 points of Integrity at the beginning of each Round until destroyed.

Shortcut Engine {500} (10)
Footprint: 10
Range: n/a
Parameter: n/a
Defense: n/a

Creates and maintains a Shortcut to the Engine's location in the deploying Program's Storage. The Shortcut functions as normal so long as the Engine remains functional.

Shortcuts

Shortcuts transport a Program from their current location to another in The Network. Shortcuts function so long as the two locations are on the same Server, or neither location's Server is locked down. They are rarely available from Vendors and more often created by Executables. Shortcuts have a Footprint of 3 if used to travel to a Subsystem on the same Server, or 7 if used to travel to another Server.

Malware

Malware is illegal and dangerous software that operates similarly to executables. When Malware is used, it is immediately removed from the Storage of the Program that used it.

Trojan {500}
Footprint: 1
Range: Adjacent
Parameter: n/a
Defense: n/a

When used, this virus attaches to a Registry item. If that item is copied to another Program's Registry, it will transmit information about a random slot in that Registry to the Hacking Program at the beginning of each Round.

Phase Condenser {1,000}
Footprint: 1
Range: n/a
Parameter: n/a
Defense: n/a

Blocks any Program on the Server from entering the targeted Port Phase for the next 6 rounds. Any Program already in the targeted Phase has their Phase set to 1 and must roll exceed 8 on a roll of 1d6 + Dash or lose 1d20 points of Integrity.

General Utility Neutralizer {5,000}
Footprint: 1
Range: Adjacent
Parameter: n/a
Defense: n/a

The hacking Program rolls their 1d6 + Shimmy, and the targeted program reduces their Integrity by four times the amount.

Appendix 3: Background Processes

Background Processes are constant effects that occur without a Program's intervention. Their constant use is a drain on the Program's resources, though, and running one reduces their Elegance by a small amount noted as Reserved Elegance.

Counter-Hacking

Adaptive Network {1,000}
Reserved Elegance: 5
When the Program is targeted by a Command, they can immediately roll their 1d6. If they roll an , they immediately increase their Phase by 1.

Recovery Process {1,000}
Reserved Elegance: 2
Restores 1 point of Integrity every 2 Ticks.

Signal Monitor {1,000}
Reserved Elegance: 1
The Program always knows if a probe, injection, or corruption Command has been used on it.

Hash Randomizer {5,000}
Reserved Elegance: 4
Adds 1d6 to the Slot number of the first Command that targets the Program in a round.

Subsystems

Corruption Enhancer {4,000}
Reserved Elegance: 5
Any Command that lowers another Program's Integrity lowers it by twice as much.

Encryption {n/a}
Reserved Elegance: n/a
Encrypts all of the Program's Data, making it indecipherable when copied unless it is decrypted first. This process generates a corresponding Decryption Key in the Program's Storage.

Mapping Matrix {1,000}
Reserved Elegance: 10
The Program can always remember any Server paths they have traveled on, and entering a new Server counts as acquiring new data for the purposes of Growth.

Protocol Trigger {3}
Reserved Elegance: 3
If this Program enters conflict, it automatically sends a flag to the Server's security protocols.

Name:

Times Crashed:

Integrity: Randomizer:

Elegance: Memory:

Storage: Processes:

Crash: Dash:

Wink: Think:

Shimmy: Shakedown:

Infosec: K-sec:

Design: _____ **Address:** ___

Name	Slot	Footprint

Compulsions **Address:** ___

Name	Slot

Kernel **Address:** ___

Name	Slot

Memory **Address:** ___

Name	Slot

Processes **Address:** ___

Name	Slot	Footprint

Name	Slot	Footprint

Storage **Address: ___**

Name	Slot	Footprint	Name	Slot	Footprint
_____	___	_____	_____	___	_____
_____	___	_____	_____	___	_____
_____	___	_____	_____	___	_____
_____	___	_____	_____	___	_____
_____	___	_____	_____	___	_____
_____	___	_____	_____	___	_____
_____	___	_____	_____	___	_____
_____	___	_____	_____	___	_____
_____	___	_____	_____	___	_____
_____	___	_____	_____	___	_____

Player Notes