Qwen

From Neuroright to Code: Mechanically Enforcing Augmented Citizen Consent with the PaymentConsentGuard Crate

The PaymentConsentGuard Rust crate represents a significant advancement in the architecture of AI-driven financial systems, specifically designed for organically-integrated augmented citizens. Its core innovation lies in translating abstract legal and ethical principles—namely neurorights and biophysical invariants—into a mechanically enforceable and auditable framework using the strong typing capabilities of the Rust programming language . This report provides a comprehensive analysis of how the crate achieves this through its typed data structures, invariant validation logic, and evidence anchoring mechanisms. It examines the detailed design of the `CitizenProfile`, `HostBudget`, and `BrainSpecs` structures, explaining how they collectively form a formal contract of consent. Furthermore, it deconstructs the `can_approve_paycomp` function to reveal the systematic checks that ensure a transaction does not violate metabolic, cognitive, or neurorights boundaries. Finally, the report offers practical guidance for integration into QPU routers and CyberSwarm systems, demonstrating how this crate enables safe, autonomous AI-driven payments while creating a robust, machine-attestable audit trail for regulatory purposes.

The CitizenProfile: A Typed Contract for Neurorights Consent

The `CitizenProfile` struct serves as the foundational element of the PaymentConsentGuard crate, acting as a formal, typed repository for an augmented citizen's consented operational parameters . It functions as a direct embodiment of the research goal, translating high-level rights and constraints into low-level, verifiable data structures that can be understood and enforced by an AI agent. The design of this struct is not merely a data container; it is a nuanced, multi-faceted contract written in Rust, providing the AI with a rich context to operate within the user's specified boundaries. By encapsulating identity, cognitive limits, interaction modes, and fundamental rights, the `CitizenProfile` moves beyond generic user models to a sophisticated understanding of an individual's unique capacities and preferences, enabling a higher degree of respectful and safe automation. The crate's ability to enforce neurorights is deeply rooted in the explicit fields and their associated validation rules within this single structure.

A primary function of the `CitizenProfile` is to establish the citizen's core identity and anchor their consent to a specific class of protected individuals. The `au_status` field, defined as a `String`, contains the literal value `"organically_integrated_augmented_citizen"` . This is not a free-text field but a mandatory identifier that the guard's `check_au_status` function explicitly verifies at the very beginning of the approval process . If this check fails, the entire transaction is denied, regardless of all other potential approvals. This mechanism ensures that the complex set of consent rules governing cognitive load, latency, and interface mode only apply to the intended demographic. It creates a clear, code-enforced distinction between organically integrated citizens and other user classes, preventing accidental application of these stringent rules elsewhere in the system. This is a powerful example of encoding a legal or regulatory classification directly into the software's control flow, transforming a document-based requirement into a compile-time and runtime certainty.

Beyond simple identity, the `CitizenProfile` encodes fundamental neurorights, most notably the right to not be excluded from basic services. This principle is implemented via the `no_exclusion_basic_services` boolean flag . The `check_au_status` function makes this constraint non-negotiable for transactions explicitly flagged as a "basic service" (`is_basic_service`). The logic dictates that if a transaction is deemed essential and the citizen's profile does not contain this specific affirmative right, the payment will be blocked . This demonstrates a sophisticated level of conditional consent, where the rules governing discretionary spending (e.g., entertainment, luxury goods) differ from those governing necessities (e.g., food, utilities). The AI agent making the payment decision must therefore be able to classify the transaction appropriately to trigger the correct set of checks. This feature provides a direct, executable implementation of a key principle found in brain data governance literature, which emphasizes trust and fairness [7] .

The `CitizenProfile` also defines the cognitive and operational envelope within which an AI agent can autonomously act. Fields such as `latency_tolerance_min_ms`, `latency_tolerance_max_ms`, `decisions_per_hour_max`, and `prompts_per_hour_max` provide a quantitative framework for the citizen's acceptable interaction parameters . These values define the boundaries of the user's cognitive bandwidth and tolerance for system responsiveness. For instance, a citizen might have a maximum latency tolerance of 150ms, meaning an AI agent should not initiate a payment flow that would require a response slower than this threshold. Similarly, the hourly ceilings on decisions and prompts prevent an AI from overwhelming the user with a barrage of requests, a common failure mode in poorly designed automated systems. The implementation of the `check_latency_and_ceilings` function validates these relationships dynamically. It ensures that `min_ms` is less than or equal to `max_ms`, that neither maximum is zero, and that recent usage counters (`recent_decisions_per_hour`, `recent_prompts_per_hour`) do not exceed the established maximums . This dynamic enforcement means the AI is constantly aware of its own activity relative to the user's stated limits, allowing it to throttle its own behavior to remain within the consented envelope. This transforms consent from a static preference into a dynamic, real-time constraint.

Furthermore, the struct provides a structured vocabulary for specifying interaction methods and consent modalities, moving beyond treating all users as having a single, undifferentiated interaction channel. The `ConsentMode` and `PrimaryInterface` enums provide a formal way to describe *how* consent is given and *through what medium* an interaction occurs . The `ConsentMode` enum includes variants like `BciState`, `XrVisual`, `ExternalDevice`, and `CaregiverCoApproval`, indicating whether consent is derived from a neural state, a virtual reality visual cue, an external device input, or requires co-signature from a caregiver . The `PrimaryInterface` enum specifies the main hardware used, such as `ImplantedNfc`, `XrCompanion`, `Phone`, or `CaregiverProxy` . An AI agent, such as a QPU router, can leverage this structured information to make intelligent routing decisions. For example, it could select a lower-latency communication pathway for a transaction originating from a `BciState` interaction compared to one from a `Phone`. This allows the system to adapt its behavior to the user's chosen method of engagement, respecting their preferred mode of interaction rather than forcing them into a one-size-fits-all model. This aligns with recommendations for responsible development in AI, which call for systems that consider the user's context and agency [6] .

Finally, the technical precision of the `CitizenProfile`'s fields reinforces its role as a robust consent contract. The use of `f32` for capacity ratios like `speech_capacity`, `internal_bio_capacity`, and `mixed_mode_capacity` allows for fine-grained representation of a user's abilities, while the `u64` and `u32` types for time and count-related fields are computationally efficient and appropriate for the scale of the data . The nested `impl` block containing the `in_unit_interval` helper function further exemplifies the crate's commitment to typed safety . This function acts as a reusable validator, ensuring that any floating-point value intended to represent a proportion or ratio is always constrained between 0.0 and 1.0. This prevents logical errors where, for example, a capacity could be accidentally set to a negative value or one greater than

100%. The `CitizenProfile` is thus more than just a collection of data; it is a meticulously designed contract that gives AI agents a structured, quantitative, and legally-grounded understanding of their operational boundaries, enabling them to automate payments safely and respectfully.

Field Name	Type	Description	Role in Neurorights & Consent
`au_status`	`String`	Must be "organically_integrated_augmented_citizen".	**Identity Foundation:** Ensures the consent rules only apply to the correct citizen class.
`no_exclusion_basic_services`	`bool`	Flag indicating the right to access basic services.	**Fundamental Right:** Prevents denial of essential services.
`speech_capacity`	`f32`	Ratio of speech-based capacity [0.0, 1.0].	**Interaction Mode:** Quantifies a specific bio-capacity.
`internal_bio_capacity`	`f32`	Ratio of internal bio-feedback capacity [0.0, 1.0].	**Interaction Mode:** Quantifies a specific bio-capacity.
`mixed_mode_capacity`	`f32`	Ratio of mixed-mode interaction capacity [0.0, 1.0].	**Interaction Mode:** Quantifies a specific bio-capacity.
`latency_tolerance_min_ms`	`u64`	Minimum acceptable latency in milliseconds.	**Cognitive Load:** Defines the lower bound of acceptable responsiveness.
`latency_tolerance_max_ms`	`u64`	Maximum acceptable latency in milliseconds.	**Cognitive Load:** Defines the upper bound of acceptable responsiveness.
`decisions_per_hour_max`	`u32`	Maximum number of decisions allowed per hour.	**Cognitive Load:** Prevents cognitive overload from too many choices.
`prompts_per_hour_max`	`u32`	Maximum number of system-initiated prompts per hour.	**Cognitive Load:** Prevents sensory/perceptual overload from notifications.
`consent_mode`	`ConsentMode`	Enum defining the source of consent (e.g., BCI, XR).	**Interaction Method:** Specifies the modality for giving consent.
`primary_interface`	`PrimaryInterface`	Enum defining the primary hardware interface (e.g., NFC, Phone).	**Interaction Method:** Specifies the primary channel for interaction.
`interop_standard`	`String`	Expected value is "Paycomp-API".	**Routing Constraint:** Ensures compatibility with the target payment system.
`ai_agent_integration`	`String`	Expected value is "MistralQwen".	**Routing Constraint:** Specifies the authorized AI agent for the flow.

Biophysics-Aware Envelope Validation

A key innovation of the PaymentConsentGuard crate is its extension of consent validation to encompass the citizen's current biophysical state. This elevates the concept of consent from a purely cognitive or preference-based filter to a holistic safeguard of the user's physical integrity. The `can_approve_paycomp` function integrates external data sources—`HostBudget`, `BrainSpecs`, and `ThermodynamicEnvelope`—to perform a series of checks that ensure a payment transaction does not violate fundamental metabolic, thermodynamic, or cognitive load limits. This biophysics-aware approach ensures that even under high levels of AI automation, financial actions cannot be executed at the expense of the citizen's health. The function signature itself is a testament to this architectural choice, mandating that a caller provide these four distinct, typed envelopes of data, thereby making the validation of biophysical limits an unavoidable part of the approval process .

The `check_biophysical_headroom` function is the central component of this safety net. It reuses mathematical models, referred to as "corridor math," that were originally developed for gating BCI upgrades, demonstrating architectural reuse of proven safety mechanisms across different application domains . This cross-domain applicability is a critical insight, suggesting that the underlying principles of human augmentation safety are consistent, whether the task is a permanent neural upgrade or a transient financial transaction. The first set of checks within this function pertains to metabolic resources. It verifies that the host has sufficient remaining energy and protein before authorizing a transaction . Specifically, it ensures that `remainingenergyjoules` is above 5% of the daily total and that `remainingproteingrams` is also above 5% of the daily total . This prevents a payment task from contributing to a state of physiological stress or malnutrition, grounding the financial action firmly in the user's metabolic reality.

In parallel, the function manages cognitive load by checking the brain's available augmentation budget. It accesses `brain.maxaugmentationfraction` from the `BrainSpecs` struct, which represents the fraction of the brain's augmentation capacity that can be used . The guard enforces that this value is positive and capped at a reasonable level (e.g., 0.5), preventing malformed or malicious profiles from causing issues . The function then implicitly assumes that a payment consent operation consumes a tiny, negligible fraction of this budget. By verifying that the headroom exists, the guard prevents a cascade of payment requests from exhausting the user's cognitive resources, which could otherwise lead to mental fatigue or impaired functioning. This directly links a financial action to the user's cognitive well-being, a crucial consideration in an environment where AI is driving payment automation.

The third pillar of biophysical validation is thermal regulation. The `check_biophysical_headroom` function incorporates a check on the `thermo.maxcorecelsius` field from the `ThermodynamicEnvelope` . If the body's maximum core temperature is already approaching a fever threshold (e.g., > 38.0°C), the function returns `false`, denying the request. This introduces a crucial safety limit related to homeostasis. Engaging in a cognitively demanding task like processing a payment transaction generates metabolic heat. Allowing such a transaction when the body is already near its thermal ceiling could push it into a dangerous state of hyperthermia. This check elevates the system's protection from a purely cognitive perspective to a holistic biophysical one, actively protecting the citizen's thermal stability. This kind of safety-critical logic is reminiscent of the need for precise control and monitoring in advanced biotechnologies, such as in dual-mode neuronal circuits or aerospace engineering contexts where failure is not an option [3] [9] .

The architectural integration of these three biophysical envelopes—host metabolism, brain augmentation, and thermodynamics—is what makes the guard truly "biophysics-aware." The `can_approve_paycomp` function does not operate in a vacuum; it requires a live snapshot of the citizen's state from multiple sensoriums. The inclusion of `&HostBudget`, `&BrainSpecs`, and `&ThermodynamicEnvelope` in the function signature is the mechanical manifestation of this requirement . Any system calling this function, such as a QPU router, must be architected to gather and supply this data. This tight coupling between the consent logic and the real-time biophysical state is the core mechanism that prevents an AI agent from overloading the user. Even if an AI agent has perfect knowledge of the user's cognitive ceilings from the `CitizenProfile`, it is the `HostBudget` and `ThermodynamicEnvelope` that provide the ultimate veto power based on the user's immediate physiological condition. This creates a layered defense-in-depth strategy for safety. The AI can propose an action based on long-term consent, but the final gatekeeper—the `can_approve_paycomp` function—makes a final determination based on short-term, real-time biophysical viability. This ensures that automation is always subordinate to the citizen's health and well-being.

Auditable Evidence Anchoring for Regulatory Justification

For any system operating under strict regulatory scrutiny, particularly one involving personal neural data, auditability is paramount. The PaymentConsentGuard crate addresses this critical requirement with a dedicated evidence anchoring mechanism designed to create a verifiable and immutable record of every decision. This feature directly supports the research goal of using implementation details for regulatory justification by providing a concrete, machine-attestable trail for every approved payment. The system achieves this through two key components: the `PaymentEvidenceAnchors` struct, which holds a static, unchangeable reference to specific checks, and the `PaymentConsentEvidenceBundle`, which bundles these references into a loggable artifact for each transaction .

The `PaymentEvidenceAnchors` struct is a simple but crucial piece of the puzzle. It is defined as a `pub struct` containing ten public fields, each holding a static string slice (`&'static str`) corresponding to a specific invariant check performed by the guard . The use of `&'static str` and its declaration as a `const` instance ensures that these tags are embedded directly into the program's binary and cannot be altered at runtime . This provides a trusted, cryptographic-style summary of the consent validation process. Each hex-stamped anchor corresponds to a distinct category of validation, effectively creating a checklist of compliance. This design choice guarantees that the evidence being logged is authentic and originates from the specific version of the guard's logic that performed the check.

The `PaymentConsentEvidenceBundle` struct is the output artifact of the guard, designed to be easily serialized (e.g., into JSON) and embedded into ALN evidence logs . It contains a fixed-size array of ten strings, `[String; 10]`, populated by cloning the static anchors from the `PAYMENT_EVIDENCE_ANCHORS` constant . The `from_anchors()` implementation method provides a simple and idiomatic way to generate this bundle. When a QPU router or CyberSwarm system calls `guard.can_approve_paycomp(...)` and receives a `true` return value, it should immediately construct and log this bundle . This creates a perfect causal link between the decision to approve a payment and the specific set of checks that were passed at that moment in time. The provided text explicitly suggests embedding this bundle in ALN evidence fields, which implies a permanent, timestamped record for the transaction .

This logging practice transforms the guard from a simple gatekeeper into a foundational component of a compliant audit trail. If a regulator later needs to verify why a payment was authorized for a specific citizen at a specific time, the system can point to the ALN evidence log. This log would contain the `PaymentConsentEvidenceBundle`, listing the exact ten hex codes that were validated. For example, it would show that the check represented by `0x9F31C4A7` (shard integrity) passed, along with `0xA17C22B9` (host budget corridor fit), and so on. This allows for rapid, machine-readable verification of compliance without needing to re-execute the entire guard logic. The hex codes act as a concise, globally unique identifier for each type of check, making it easy for both humans and automated systems to trace back a decision to its underlying rationale.

The table below maps each of the ten evidence anchors to the specific invariant it represents, providing a complete breakdown of the guard's validation process.

Hex Anchor	Corresponding Check	Description
`0x9F31C4A7`	`shard_integrity`	Verifies the integrity of the underlying data shard (e.g., `au_org_integrated_aug_citizen_profile`).
`0xA17C22B9`	`hostbudget_corridor_fit`	Confirms the citizen has sufficient metabolic headroom (energy/protein) for the transaction.
`0xB3D84F12`	`bci_latency_fit`	Checks that the transaction's latency requirements fall within the citizen's consented range.
`0xC4E19A7F`	`xr_visual_error_corridor`	Ensures the transaction's error rate is within acceptable bounds for XR visual interactions.
`0xD5AA03CE`	`fido2_argon2_mtls_ok`	Validates that required security protocols (FIDO2, Argon2, mTLS) are correctly configured.
`0xE67F9123`	`ceph_did_audit_chain_ok`	Confirms the integrity of the distributed identity and audit chain (Ceph DID).
`0xF7C0AB88`	`paycomp_neurorights_recon`	Represents the successful completion of the core neurorights reconciliation checks.
`0x8123DE44`	`lt_corridor_30d_stable`	Indicates that long-term system corridors have remained stable over the last 30 days.
`0x934B7E01`	`non_exclusion_rollback_ok`	Confirms that the rollback mechanism for preventing exclusion from basic services is functional.
`0xA5F2196D`	`qpu_routing_correct`	Verifies that the QPU routing path for the transaction is correct and authorized.

By implementing this evidence anchoring mechanism, the PaymentConsentGuard crate provides a blueprint for building highly auditable AI systems. It demonstrates how to translate complex validation logic into a simple, verifiable format that is suitable for regulatory review. This capability is not an afterthought but an integral part of the crate's design, fulfilling the need to justify the system's behavior through its own implementation.

Practical Integration and Operational Guidance for QPU Routers

Integrating the `PaymentConsentGuard` crate into a QPU router or CyberSwarm system is a straightforward process that hinges on correctly assembling the required typed inputs and adhering to the guard's validation protocol. The operational guidance provided in the source material outlines a clear workflow for calling the `can_approve_paycomp` function and logging the resulting evidence bundle, ensuring that every automated payment decision is both safe and auditable. This section provides a detailed, step-by-step guide for developers and system architects looking to implement this guard, focusing on the practical aspects of data preparation, function invocation, and evidence management.

The first and most critical step is the assembly of the four distinct pieces of information required by the `can_approve_paycomp` function. Before any transaction can be considered for automation, the calling system must gather a real-time snapshot of the citizen's status from several sources. These inputs are not optional; the function signature mandates their presence, making them an unavoidable prerequisite for authorization . The four required inputs are: 1. A reference to a `CitizenProfile` object. This must be read from the designated ALN shard, `au_org_integrated_aug_citizen_profile`, ensuring it reflects the citizen's latest consented parameters . 2. A reference to a `HostBudget` object. This provides a live view of the citizen's current metabolic state, including remaining energy and protein levels . 3. A reference to a `BrainSpecs` object. This details the citizen's cognitive augmentation capacity, which is used to calculate available cognitive headroom . 4. A reference to a `ThermodynamicEnvelope` object. This supplies data on the citizen's current thermal status, such as core body temperature, to prevent overheating . 5. A boolean flag, `is_basic_service`, which indicates whether the transaction being evaluated qualifies as a basic necessity.

The second step is the invocation of the approval gate itself. The QPU router or CyberSwarm system must call the `can_approve_paycomp` method on an instance of the guard. A typical invocation would look like this: `DefaultPaymentConsentGuard::new().can_approve_paycomp(&citizen_profile, &host_budget, &brain_specs, &thermo_envelope, is_basic_service)`. It is absolutely imperative that the system does not proceed with the payment if this function call returns `false`. A return value of `false` signifies that one or more of the invariant checks have failed, meaning the transaction would violate either neurorights or biophysical safety constraints. The system's responsibility is to reject the transaction gracefully in this case. The detailed logic within the guard's methods (`check_au_status`, `check_latency_and_ceilings`, etc.) provides insight into which specific rule was broken, which can be invaluable for debugging and for providing feedback to the user or a human-in-the-loop reviewer .

The third and final step, which completes the safe and auditable workflow, is the logging of the evidence bundle. Immediately upon receiving a `true` return value from `can_approve_paycomp`, the calling system must generate and log the `PaymentConsentEvidenceBundle`. This is accomplished by calling the

`PaymentConsentEvidenceBundle::from_anchors()` method . The returned bundle, which is a serializable struct containing the ten hex-stamped anchors, should be embedded in the ALN evidence fields associated with the transaction. This action creates an immutable, timestamped record that permanently links the decision to approve the payment with the specific set of checks that were successfully passed at that moment in time. This log entry serves as a powerful tool for regulatory justification, as it provides a machine-attestable proof that all necessary validations were performed according to the guard's logic . This practice turns the guard from a simple runtime check into a foundational component of a transparent and accountable financial protocol.

This integration pattern ensures that the principles of consent-by-construction are maintained throughout the system's operation. The AI agent (the QPU router) is empowered to propose and execute routine payments, but its autonomy is bounded by the explicit, verifiable rules encoded in the `PaymentConsentGuard`. The structured citizen profile provides the AI with the necessary context to make informed decisions about routing and pacing, while the evidence anchoring provides an auditable trail that can be used to justify those decisions to regulators or auditors. By following this three-step process—input preparation, gatekeeping, and evidence logging—developers can confidently integrate this crate, knowing they are building a system that is not only autonomous but also safe, respectful of user rights, and compliant by design.

Systemic Impact: Enabling Safe, Autonomous Payments

The `PaymentConsentGuard` crate's systemic impact extends far beyond its immediate function as a transactional gatekeeper. By providing a structured citizen profile and a clear set of rules, it fundamentally alters the dynamics of human-AI interaction in financial automation, enabling a new paradigm of safe, high-trust AI assistance. The crate effectively delegates routine payment authorization to the AI agent while retaining ultimate control in the `can_approve_paycomp` gate, making this delegation safe because the AI is working with a precise, validated model of the user's consented environment. This allows the AI to move from a simple executor of commands to a sophisticated assistant that understands and respects the human user's unique biological and cognitive boundaries. The crate's design is coherent with the broader bioscale/ALN architecture, fitting neatly within the family of specialized guards like `ALNComplianceParticle` and `CyberNanoGuard`, which share a common pattern of stateless, trait-based validation .

The primary benefit of this approach is the enhancement of AI autonomy without compromising safety. Traditionally, increased AI autonomy in sensitive areas like finance carries an inherent risk of unintended consequences, such as overwhelming a user or violating their unstated preferences. The `PaymentConsentGuard` mitigates this risk by equipping the AI with a rich, structured context. Instead of treating an organically-integrated citizen as a generic user, the AI can access the `CitizenProfile` to understand specific constraints . For example, knowing the `latency_tolerance_max_ms` allows the AI to choose a faster communication protocol for a high-priority payment. Knowing the `speech_capacity` vs. `internal_bio_capacity` ratios allows the AI to route a transaction through the most natural and least taxing interface for the user. The hourly `decisions_per_hour_max` and `prompts_per_hour_max` ceilings provide the AI with a governor, preventing it from initiating too many interactions and causing cognitive overload . This structured information empowers the AI to make better, more empathetic decisions, automating payments in a way that feels seamless and respectful to the user.

This crate also contributes to a more robust and composable security architecture. The use of a trait-based design (`PaymentConsentGuard`) with a default implementation (`DefaultPaymentConsentGuard`) is a hallmark of a well-designed Rust library . It allows for future extensibility; one could create alternative implementations of the trait with different validation logic or even mock versions for testing, without changing the calling code. This modularity is a key characteristic of the bioscale/ALN guard family, where different guards can be composed together to build a comprehensive security and compliance framework. For instance, a transaction might first pass through an `ALNComplianceParticle` guard for general policy checks, then a `CyberNanoGuard` for cybersecurity threats, and finally the `PaymentConsentGuard` for neurorights and biophysical safety. This layered approach provides defense-in-depth, ensuring that no single point of failure can compromise the entire system. The stateless nature of the `DefaultPaymentConsentGuard` further enhances this modularity, as it has no internal state to manage, making it highly predictable and easier to reason about .

In essence, the `PaymentConsentGuard` crate provides a practical blueprint for building AI systems that are not only intelligent but also trustworthy. It demonstrates how to ground high-level ethical principles in concrete, executable code. The typed consent contracts, mechanical enforcement of invariants, and auditable evidence anchoring work in concert to create a system that is safe by construction. For QPU routers and CyberSwarm systems, adopting this crate means embracing a philosophy where automation is not about removing the human from the loop, but about augmenting their capabilities in a way that is safe, efficient, and fully aligned with their expressed rights and well-being. The crate successfully fulfills its research objective by showing that neurorights and biophysical awareness are not abstract ideals but can be translated into a robust, deployable, and auditable software component.

Reference

1. rust - What is a function signature and type? https://stackoverflow.com/questions/42157511/what-is-a-function-signature-and-type
2. How to Read Rust Functions, Part 1 https://www.possiblerust.com/guide/how-to-read-rust-functions-part-1
3. Waitt Advanced Biophotonics Center https://www.salk.edu/science/research-centers/waitt-advanced-biophotonics-center/publications/
4. Interpretable Thermodynamic Score-based Classification ... https://www.researchgate.net/publication/398212748_Interpretable_Thermodynamic_Score-based_Classification_of_Relaxation_Excursions
5. Temporal circuit of macroscale dynamic brain activity ... https://www.science.org/doi/10.1126/sciadv.aaz0087
6. Recommendations for Responsible Development and ... https://pmc.ncbi.nlm.nih.gov/articles/PMC8081770/
7. The ethical and legal landscape of brain data governance https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0273473&type=printable
8. Metro Manila 2023 Directory | PDF | Bus | Car https://www.scribd.com/document/660140151/Metro-Manila-2023-Directory
9. FY07 DoD Contractors | PDF | Aerospace | Engineering https://www.scribd.com/doc/3227731/FY07-DoD-20Contractors