BRC-100 Risk Assessment (v2.0)

Centralization, Surveillance, Security and Ecosystem-Control Risks (Pgs 1 – 8); Best Integration Practices Guide (Pgs 9 - 22)

BRC-100 is a powerful integration tool. It offers applications a common interface through which they can interact with compatible BSV wallets, construct transactions, manage outputs, obtain cryptographic keys, request identity certificates, discover identities, encrypt data, create signatures and reveal key relationships. **With that power, however, come enormous centralization risks affecting privacy, security, financial autonomy and the future structure of the BSV ecosystem.**

This report is intentionally risk-focused. It does not allege that every capability described below is currently being abused. It asks whether widespread adoption of BRC-100 helps create an ecosystem that could become vulnerable to surveillance, coordinated exclusion, legal compulsion, application gatekeeping, infrastructure failure or loss of independent control. The central question is:

> **Why must a payment or token application inherit a comprehensive identity-and-audit-capable wallet architecture merely to obtain wallet interoperability?**

Human and machine use cases should not be conflated. **BRC-100's broad, structured interface may offer its greatest practical rewards to AI agents** and autonomous software that must interact with numerous applications, counterparties and paid services without repeated human intervention. **That does not establish that the same architecture is the best default for human** savings, identity, credentials or long-term financial control. An interface optimized for machine autonomy can impose disproportionate risks when it becomes the gateway to human-owned funds and personal information.

Below, we outline the principal risk categories to the best of our ability.

1. Concentration of Financial, Identity and Application Power

BRC-100 does far more than standardize the sending of payments. Its interface encompasses transaction creation, transaction and output listing, output baskets, master and derived public keys, encryption, signatures, identity certificates, identity discovery and two forms of key-linkage revelation. Every request identifies the originating application by its fully qualified domain name, which the specification says may be used for permissions, access control and audit logging.

This consolidates functions that could otherwise remain separated. Money, application permissions, transaction history, identity, credentials and cryptographic relationships may all pass through one common BRC-100 wallet layer. That wallet therefore becomes both a high-value attack target and a potential control point over a large portion of the user's digital activity.

Compartmentalization is often mistaken for needless fragmentation. Separate payment, social, gaming, business, identity and savings wallets may be less convenient, but they prevent one compromised

application, software dependency or permission decision from reaching everything else. BRC-100's broad interoperability model risks replacing those compartments with a single, highly consequential gateway.

2. Institutional Standardization and Ecosystem Monoculture

BRC-100 is not a Bitcoin consensus rule. Applications can still create valid BSV transactions without it, and legacy P2PKH transactions remain valid. Nevertheless, BRC-100 is being presented through official BSV documentation, development examples, SDK tooling and Wallet Toolbox implementations as the standard interface through which applications should interact with wallets. Official demonstrations increasingly build around BRC-100, BRC-29, identity keys, Wallet Toolbox and BSV Desktop.

This can create a self-reinforcing adoption cycle. Institutional tooling makes BRC-100 the easiest supported path; developers follow the supported path; applications begin expecting BRC-100; and wallets that do not adopt it become commercially disadvantaged. The resulting network effect can then be cited as proof that BRC-100 was the inevitable or technically superior standard.

"Vendor-neutral" does not necessarily mean architecture-neutral or governance-neutral. Different wallet brands may implement the same interface while depending on the same specifications, SDKs, Wallet Toolbox components and underlying design philosophy. A vulnerability, policy change or controversial architectural assumption can therefore spread across nominally independent products.

The centralization risk is not that one organization directly controls every wallet. It is that one institutionally favored architecture may become the practical price of admission to the useful BSV application ecosystem.

3. Identity and Certificate Gatekeeping

BRC-100 identity and BRC-52 certificates are separate. The identity key is the wallet's master public identity key. A certificate is a signed claim issued by a certifier that links that subject key to particular attributes. Revoking a certificate does not destroy the identity key, erase the wallet or automatically freeze funds.

Nevertheless, the certificate system creates a serious gatekeeping risk. BRC-52 certificates can contain verified identity attributes, name the certifying entity and include a revocation outpoint. If that outpoint is spent, verifiers should treat that certificate as revoked.

The danger emerges when applications make a particular certificate mandatory. If many important applications require the same certificate type, accept only a narrow group of certifiers or adopt shared trusted-certifier defaults, revocation of one certificate could exclude a person from a substantial portion of the ecosystem. The person's cryptographic identity would technically continue to exist, but it could become commercially useless.

This creates the possibility of KYC creep. Certificates may begin as optional tools for regulated services, age restrictions or professional credentials. Over time, applications may require them because

they simplify fraud prevention, compliance or account recovery. Pseudonymous participation can then become second-class participation without any formal protocol rule banning it.

4. Behavioral Tracking and Metadata Surveillance

Every BRC-100 request carries the originating application's domain. This permits a BRC-100 wallet to know which application is requesting a payment, key, signature, certificate, basket or other operation. The specification expressly identifies audit logging as a use for this originator information.

The specification does not require every implementation to preserve a permanent centralized record of every request. The risk is that the information necessary to create such a record is structurally present. A wallet implementation, remote storage provider, compromised device or forensic examiner may be able to reconstruct a detailed map of the user's application relationships and financial behavior.

Current BRC-100 implementations demonstrate the importance of transaction-history storage. Yours Wallet, for example, states that it tracks both keys and transaction history to locate assets and supports local storage, active remote storage and remote backups.

Private keys remaining on the user's device do not eliminate this exposure. A provider may never possess the signing key while still possessing transaction histories, application metadata, output records, device information or authentication logs. This creates a form of informational custody in which the provider cannot directly spend the money but may hold much of the information needed to understand, locate or recover it.

The BSV Association itself describes BSV as traceable, regulatory-friendly and not designed for anonymity. Once a real-world identity is linked to any relevant key or output, public-ledger analysis can be combined with wallet, application, certificate and storage records.

5. Subpoenas and Coercible Infrastructure

A fragmented system forces investigators to identify and approach multiple independent applications, wallet systems and service providers. A standardized architecture can create clearer and more valuable legal targets.

Depending upon implementation, those targets may include wallet vendors, remote storage providers, authentication services, application operators, identity certifiers and overlay or lookup services. They may hold different portions of the user's identity, transaction history, app-usage information, certificate records and output metadata.

BRC-100 does not create a single central database, and a self-hosted implementation may eliminate some third-party targets. The centralization risk is that consumer convenience will generally favor a small number of default wallets, hosted storage systems, certifiers and application providers. Legal demands directed at those few entities could reveal information spanning multiple applications rather than one isolated service.

Selective disclosure limits what a certificate holder initially reveals. It cannot force a verifier to forget information after disclosure, prevent compelled production of stored information or stop regulated applications from making broader disclosure a condition of service.

6. Auditability and Key-Linkage Revelation

BRC-100 includes `revealCounterpartyKeyLinkage` and `revealSpecificKeyLinkage`. The first can reveal the relationship between the user and a counterparty across their interactions; the second can reveal the relationship involving a specific protocol and key. The BRC-100 documentation describes these capabilities as supporting identity verification and auditing.

These revelations are not automatically public. They are encrypted for a verifier and should be subject to wallet permissions. The risk is the normalization of an audit capability that may later be demanded by employers, financial institutions, platforms, regulators, litigants or governments.

Consent can also become nominal rather than meaningful. A user may technically approve a revelation while having little realistic ability to refuse a required application, account, employment condition or legal demand. Complex permission prompts increase the likelihood that users will not understand whether they are revealing one transaction, one derived key or an entire counterparty relationship.

Output baskets, labels and tags are not automatically published to the blockchain as a financial report. But if a BRC-100 wallet database or remote storage record is obtained, this internal organization may make financial reconstruction substantially easier than analyzing raw outputs without context.

7. Convergence with Digital Asset Recovery

BRC-100 does not itself implement Digital Asset Recovery, and certificate revocation does not freeze BSV. These are separate systems.

The risk lies in their potential convergence. BRC-100 wallets and associated services can create records linking identities, applications, payment relationships, transaction histories and particular outputs. Digital Asset Recovery is presented by the BSV Association as a process through which participating miners comply with valid court orders involving assets on the BSV ledger.

An investigator must still establish that particular outputs belong to the targeted person. BRC-100 does not automatically deliver that mapping to miners or a recovery authority. However, identity certificates, wallet records, application logs, key-linkage revelations and remote storage data could reduce the cost of making that connection.

The resulting systemic risk is a potential identification-to-enforcement pipeline: first identify the person, then correlate the person with applications and derived keys, then identify relevant outputs, and finally invoke whatever lawful freezing or reassignment mechanism is available. Each layer may be independently defensible, but their combination creates far greater enforcement capability than any one layer alone.

8. Hot-Wallet Security and Permission Failure

A BRC-100 wallet is designed to interact continuously with applications. Applications may request transactions, keys, signatures, encryption operations, certificates, output information and spending authorization. This makes the ordinary BRC-100 wallet fundamentally different from cold storage that has no general-purpose application interface.

The standard includes permission controls and a privileged key mode, but it does not itself provide a complete high-value custody architecture involving air-gapped signing, mandatory hardware isolation, multisignature approvals, withdrawal delays or independent treasury devices.

This helps explain recurring warnings that application-connected BRC-100 wallets should hold only small operating balances. Such advice is not a protocol limit; it is an acknowledgment of the expanded attack surface. A universal wallet architecture that developers consider appropriate only for pocket money leaves ordinary users without an equally promoted system for holding substantial amounts safely.

Fine-grained permissions can reduce risk, but they also create permission fatigue. Users cannot realistically evaluate every protocol ID, basket, counterparty, certificate field or privileged request. Repeated prompts train users to approve requests reflexively, while persistent permissions allow an application trusted once to retain capabilities later.

9. AI Agents Operating with Human Funds

BRC-100’s strongest interoperability benefits may arise when an AI agent must transact with many applications and services automatically. A structured wallet interface can allow an agent to create transactions, spend within approved limits, use derived keys, access baskets, sign messages, disclose certificate fields and establish relationships with new counterparties without requiring the human owner to manually approve every individual interaction.

That efficiency creates a new class of risk. A compromised, manipulated or poorly aligned agent could operate against human-owned funds at machine speed while remaining technically within permissions the user previously granted. The failure could originate in the agent itself, an adversarial prompt, a compromised application, an external model provider, a software update or an incorrect interpretation of the user’s instructions.

The human may believe that a spending authorization defines the purpose for which money can be used. The BRC-100 wallet may be able to enforce only the technical amount, protocol, basket, counterparty or time limit encoded in the permission. It cannot reliably determine whether the agent’s decision serves the human’s true intent.

Small spending limits do not eliminate the danger. An agent may make repeated micropayments, accumulate losses over time, disclose sensitive relationships, sign harmful messages, interact with fraudulent services or reveal certificate information without any single transaction appearing large enough to trigger concern.

The risk becomes especially severe when the AI agent’s operating wallet is also the human user’s primary BRC-100 wallet, identity store or token repository. A system designed to give software broad interoperability should not automatically be given access to the human owner’s complete financial and identity environment. BRC-100 can therefore become not merely an application gateway, but a machine-operated control layer over human assets.

10. Recovery, Exit and Legacy Lock-In

BRC-100 expressly rejects reliance on legacy BIP32 key derivation and requires migration to BRC-42/BKDS. BRC-75 retains mnemonic words but derives a single master private key differently from conventional BIP32/BIP44 wallets. Entering the same words into a different wallet architecture therefore does not necessarily reproduce the same keys or balance.

Ordinary BSV is not cryptographically trapped inside BRC-100. While the original BRC-100 wallet remains functional, it can construct a conventional P2PKH output and send the BSV to an address controlled by a legacy wallet.

The greater risk is disaster recovery. BRC-29 payments depend upon the sender identity key and payment-specific derivation prefixes and suffixes supplied with the payment message. A replacement BRC-100 wallet may therefore require not only the master key but also sufficient transaction and derivation records to rediscover and reconstruct every spendable output.

BRC-100 standardizes the application-to-wallet interface. It does not by itself guarantee complete cross-vendor database portability, seed-only recovery or an independent emergency sweep into a legacy wallet. If the original BRC-100 wallet, storage provider or proprietary backup disappears, a user may retain cryptographic ownership while losing practical access to the information needed to locate or derive all relevant keys.

This creates the possibility of a functional one-way migration: funds can enter the BRC-100 environment easily, but complete exit may depend upon the continued operation and export capabilities of the original implementation. Tokens and application assets face still greater lock-in because their value may depend upon custom scripts, baskets, overlays and application-specific state that a legacy wallet cannot preserve.

Legacy addresses may remain valid forever at the blockchain level while becoming increasingly irrelevant at the application level. That is practical exclusion without formal prohibition.

11. Overbroad Interoperability

The official BRC catalog contains separate standards for Direct Payment Protocol, Paymail, BRC-29 payments, certificates, output baskets and wallet communications. This demonstrates that payment interoperability does not technically require every form of wallet, identity, data and audit interoperability to be bundled together.

Applications may need the ability to receive payment or recognize a token. They do not necessarily need access to the user's master identity, certificates, transaction history, unrelated outputs, encryption system or audit relationships.

Maximum interoperability is not automatically desirable. Payment interoperability can reduce lock-in. Identity interoperability can increase correlation. Token interoperability can improve portability. Universal wallet interoperability can enlarge the blast radius of compromise. Audit interoperability can simplify compliance while also simplifying surveillance.

The appropriate security principle is the minimum interoperability required for the application —not the broadest interface the ecosystem can create.

12. Other Considerations

Standardized attack surface. A common public interface allows security review, but it also allows malicious applications, phishing systems and exploit developers to target many compatible wallets through the same methods and permission concepts.

Implementation monoculture. Multiple branded wallets built upon common SDK and Wallet Toolbox components may share vulnerabilities and architectural assumptions. Apparent vendor diversity can conceal dependency concentration.

Overlay dependence. Tokens, identities and application state may rely upon overlay and lookup services to locate and interpret outputs. Failure, censorship or consolidation of those services can make valid blockchain data practically unusable.

Application deplatforming. BRC-100 permissions may protect users from applications, but dominant wallet implementations can also refuse particular applications, protocols, certifiers or counterparties. Ecosystem access may therefore shift from open transaction validity toward wallet-mediated approval.

Certifier concentration. A certificate system may be technically open while the market consolidates around a few recognized certifiers. Applications accepting only those certifiers create privately administered participation requirements.

Compliance creep. Optional identity and audit tools can become expected defaults because they reduce institutional risk. Once broadly deployed, the absence of a certificate or refusal to reveal linkage may itself be treated as suspicious.

Unchanging-interface risk. The promise of an unchanging standard creates stability, but it can also make foundational mistakes difficult to correct. Extensions and workarounds may accumulate around an architecture that applications have become economically unable to leave.

Strategic dependency. Applications that deeply integrate BRC-100 inherit dependence upon BSV-specific standards, tooling, wallets and network effects. If adoption stalls or the favored architecture changes, migration costs may be substantial.

Conclusion

BRC-100's greatest risk is not any single method. It is the convergence of money, identity, certificates, permissions, applications, transaction history, output classification, storage, auditability, autonomous software and legal enforceability within one institutionally favored architecture.

None of these components automatically creates centralized control. Together, however, they can make centralized observation, exclusion, coercion and enforcement easier. They can also make it possible for applications or AI agents to operate against human-owned assets through previously granted permissions that the human no longer meaningfully supervises. **A system may remain decentralized at the blockchain-consensus layer while becoming highly centralized at the wallet, identity, application and service layers through which ordinary people actually use it.**

BRC-100 should therefore be evaluated not merely as a convenient wallet API, but as a proposed operating architecture for much of the BSV application ecosystem. Its adoption should not be treated as inevitable, and developers should not be required to inherit identity-and-audit capabilities merely to obtain basic payment or token interoperability. An architecture that is highly advantageous for autonomous software should not automatically become the default architecture for human savings, identity and financial life.

This is not to say that BRC-100 offers no rewards. Some developers may decide that its capabilities justify the risks, while others may need to implement it because of specific application, customer, regulatory or compatibility requirements. For those situations, a separate **BRC-100 Best Integration Practices Guide** has been prepared to address how implementation choices can limit—but not eliminate—the risks identified in this report.

[See following Pages for Best Integration Practices Guide]

BRC-100 Best Integration Practices Guide

Risk-Limiting Recommendations for Applications, Wallets and AI Agents

Before implementing BRC-100, developers should pause and determine whether their application or wallet genuinely requires its extensive transaction, identity, certificate and interoperability capabilities. Institutional momentum, ecosystem pressure and fear of exclusion should not substitute for an independent assessment of necessity, security and long-term consequences.

BRC-100 may be especially valuable for machine-to-machine commerce and AI agents, but that does not establish that it is the appropriate default for every human-facing payment, token, wallet or social application. Once an ecosystem becomes dependent upon a common wallet architecture, reversing that decision may be extremely difficult.

This guide is not an endorsement of universal BRC-100 adoption. It is intended for developers who, after considering the risks and alternatives, nevertheless determine that BRC-100 is necessary for their product.

No implementation can eliminate all of the risks identified in the accompanying **BRC-100 Risk Assessment**. The objective is to reduce exposure through strict compartmentalization, minimal permissions, independent recovery, legacy compatibility and a clear separation between human wealth, human identity and application-controlled activity.

The Two Sides of the BRC-100 Boundary

BRC-100 defines an interface between two distinct actors:

The application side initiates requests. An application may ask the user’s BRC-100 wallet to create a transaction, access a protocol or basket, provide a derived key, disclose an identity key or certificate, sign or encrypt data, or reveal a key relationship.

The wallet side receives and mediates those requests. The wallet authenticates the application origin, determines whether permission exists, presents consent prompts, grants or denies access, signs transactions, manages wallet records, stores permissions and provides recovery and exit functions.

These responsibilities should not be confused.

A wallet supporting a BRC-100 capability does not mean every application should receive it. An application requesting a capability does not mean the wallet should automatically grant it. A safe implementation requires restraint on both sides of the boundary.

Some companies may operate on both sides—for example, by providing a BRC-100 wallet while also operating applications that connect to it. Each role should still be evaluated and disclosed separately.

Executive Implementation Standard

Before a BRC-100 application or wallet is trusted with meaningful user funds, its developers should be able to answer **yes** to each applicable question:

1. **[Application]** Is BRC-100 genuinely necessary for the application’s core function?
2. **[Application]** Does the application request only the minimum methods and permissions required?
3. **[Application]** Is the root identity key avoided whenever an application-specific derived key will work?
4. **[Application]** Are identity certificates optional unless a verified attribute is genuinely required?
5. **[Application]** Are certificate requests limited to the minimum necessary fields?
6. **[Wallet]** Are spending permissions small, short-lived, cumulative and immediately revocable?
7. **[Wallet]** Is the application origin authenticated rather than merely accepted as a claimed domain?
8. **[Wallet]** Are permissions isolated so one application cannot access another application’s protocols, baskets or records?
9. **[Wallet]** Is local storage available and preferred by default?
10. **[Wallet]** Can users export every record required to recover their BSV and supported assets?
11. **[Wallet]** Has recovery been tested without the original wallet installation, vendor or storage provider?
12. **[Wallet]** Can ordinary BSV be sent directly to a conventional legacy address?
13. **[Both]** Can supported tokens and application assets move to an independent implementation?
14. **[Both]** Are AI agents isolated in separate wallets with externally enforced limits?
15. **[Wallet]** Can users immediately revoke an application or agent while preserving access to their funds?
16. **[Both]** Have supported capabilities, software dependencies, storage practices, logging and central service providers been publicly disclosed?
17. **[Both]** Does any expansion of access require renewed consent and visible notice?

If the answer to an essential question is no, the implementation should remain experimental and limited to disposable balances.

1. Determine What the Product Actually Needs

Do not begin by asking how much of BRC-100 can be integrated. Begin by defining the product’s minimum functional requirements.

An application may need only to request payment and confirm receipt. A token application may need access to one specific protocol and output basket. A regulated service may need proof of one certified attribute. An AI agent may require a small operating balance and tightly restricted transaction authority.

A wallet may choose to support a broad range of BRC-100 capabilities, but it should not enable or approve them indiscriminately.

Before development begins, classify every proposed capability as:

Required: The product cannot perform its essential function without it.

Optional: The capability improves convenience but is not necessary.

Prohibited: The capability creates more risk than value and will not be requested or automatically granted.

The application should then adopt the narrowest possible integration profile.

A **payment-only application** should use only the transaction and payment functions it requires. It should not request the root identity key, certificates, discovery, broad output history or linkage revelation.

A **token application** should access only the specific protocol and basket required for that token. It should not inspect or spend unrelated outputs.

An **authentication application** should use an application-specific derived key rather than the wallet's persistent identity key.

A **regulated credential application** should request only the exact certificate type and fields required for the stated purpose.

An **AI-agent application** should operate through a dedicated wallet, identity and fixed budget created specifically for that agent.

The application's internal architecture should not become inseparable from BRC-100. Developers should build a narrow adapter layer so BRC-100 remains one supported interface rather than the permanent foundation of the application's user accounts, identity records and asset databases. This preserves the ability to support conventional payments, Paymail, another wallet provider or a future competing interface.

Wallet developers should likewise avoid assuming that every supported method belongs in every user flow. Broad wallet capability should be paired with narrow application access.

2. Separate Human Wealth, Application Funds and Agent Funds

A human user's principal savings should not reside in the same BRC-100 wallet that continuously accepts requests from applications.

The **BRC-100 operating wallet** should contain only the funds needed for near-term activity. The appropriate limit is not one universal dollar amount; it is the amount the user can tolerate losing if the wallet, connected application, permission manager, device or software supply chain fails.

Higher-value holdings should remain in a separate **reserve wallet** outside the ordinary BRC-100 application interface. That may be a conventional wallet, hardware-isolated wallet, multisignature

arrangement or another custody system designed for long-term value. It should replenish the operating wallet without revealing its seed, private keys or signing authority to the BRC-100 environment.

The operating wallet must not have automatic, unlimited replenishment authority. Otherwise, an application or agent that exhausts its operating balance can simply draw additional funds from the reserve, defeating the purpose of the limit.

At minimum:

- The reserve wallet should expose no general-purpose application interface.
- The operating wallet should have a defined maximum balance.
- Replenishment should require a separate human action.
- High-value tokens and long-term credentials should remain outside the operating wallet unless absolutely necessary.
- Wallet providers should clearly state that the application-connected wallet is not intended to hold the user's total savings.

AI agents require an additional boundary. Each agent—or each materially different agent role—should receive a separate BRC-100 wallet with separate keys, identity, storage, permissions and funds. Creating another basket inside the human owner's main wallet is not adequate isolation.

An agent wallet should have a fixed maximum balance, per-transaction cap, cumulative hourly and daily limits, rate limits, approved protocols, approved counterparties, expiring permissions and a manual emergency stop. It should have no automatic access to reserve funds, no root human identity key, no human certificates by default, no key-linkage authority and no ability to change its own limits.

The **application side** should request only the authority required for the agent's defined role. The **wallet side** should enforce the hard limits independently of the agent and independently of any application the agent controls.

Natural-language instructions are not financial controls. An AI agent may misunderstand the user, be manipulated through prompt injection or pursue a nominal objective in a harmful way. Hard financial and identity limits must therefore be enforced outside the AI model and outside any software layer the agent can modify.

3. Minimize Identity and Certificate Exposure

The wallet's root identity key should not become the default identifier for every application.

Ordinary authentication should use an application-specific derived key. That identifier should be unique to the defined application context, separate from payment keys, replaceable if the relationship ends and unusable for discovering the user across unrelated services.

An application should not request the root identity key unless it can document why a derived key is insufficient. It should not use the root identity key as the primary database key for a social network, game, merchant account or ordinary payment service. Doing so transforms one application breach into a cross-application correlation event.

A wallet should require clear, specific approval before revealing the root identity key. Identity-key permission should not be inferred from ordinary login, payment or account creation.

Identity certificates should remain optional unless a verified attribute is genuinely required. A certificate is a signed claim associated with the user's identity key; it is not the identity key itself. Certificate revocation may invalidate that claim and consequently block access to applications that require it.

Best practice requires:

No certificate by default. Ordinary payments, posting, gaming and token transfers should remain pseudonymous unless identity is essential.

Minimum disclosure. Request "over 18," not date of birth; "eligible jurisdiction," not full home address; "authorized employee," not unrelated personnel information.

Purpose limitation. Do not reuse one certificate across unrelated applications merely because it is convenient.

Multiple certifiers. Accept equivalent credentials from more than one qualified certifier wherever possible.

Visible trust rules. Wallets and applications should disclose which certifiers are accepted, who controls that list and how it may change.

Clear revocation consequences. Applications should tell users exactly which functions will become unavailable if the certificate is revoked.

Alternative verification. Where legally possible, provide another way to establish eligibility.

Applications should avoid identity-discovery methods unless public discovery is central to the application and clearly disclosed. Wallets should treat discovery requests as separate identity-sensitive operations, not as ordinary account access.

4. Apply Strict Permission and Origin Controls

Safe permission handling requires different duties on each side of the boundary.

Application-Side Responsibilities

Every requested permission should correspond to a specific feature the user can understand.

Applications should:

- request the smallest practical amount and shortest duration;
- use separate protocol IDs for signing, encryption, authentication and payments;
- request access only to their own basket namespace;
- avoid broad transaction-history or output access;
- request permissions when the relevant feature is first used rather than during a broad onboarding bundle;

- avoid requesting identity keys, certificates or linkage functions for unrelated convenience; and
- explain why each sensitive capability is required.

An application should not treat wallet support for a method as permission to request it routinely.

Wallet-Side Responsibilities

The wallet should independently determine whether a request is properly authenticated, permitted, understandable and proportionate.

Wallets should enforce per-transaction limits as well as cumulative hourly, daily and monthly spending limits. A monthly limit alone may allow an application or agent to spend the entire allowance immediately.

Fresh approval should be required when:

- the amount exceeds normal use;
- the recipient is new;
- the payment pattern changes;
- the destination changes after presentation;
- a recurring authorization renews;
- an application requests a new protocol or basket;
- identity or certificate access expands; or
- linkage revelation is requested.

Grouped permission requests should not become a method of acquiring broad consent during onboarding. Permissions should expire by default. Indefinite access should require an explicit and unusual decision.

The wallet should provide a clear permissions dashboard showing:

- application name;
- authenticated origin;
- approved protocols;
- basket access;
- identity-key access;
- certificate fields;
- linkage permissions;
- spending limits;
- expiration dates;
- last use; and
- immediate revocation controls.

Revoking one application must not interfere with unrelated applications or the user's ability to access and transfer their funds.

Authenticate the Originator, Not Merely the Claim

The originator field is useful only if the wallet verifies that the request genuinely came from the claimed application.

The wallet must not trust a domain value merely because an application supplied it. Implementations should defend against forged originator headers, DNS rebinding, malicious localhost requests, cross-origin attacks, compromised extensions, application impersonation and development settings accidentally carried into production.

Permissions must be bound to an authenticated origin. Similar-looking domains, internationalized-domain tricks, certificate changes and unexpected origin changes should trigger prominent warnings and require new approval.

5. Preserve Exit, Recovery and Wallet Independence

Every consumer BRC-100 wallet should provide a visible and tested method for sending ordinary BSV to any valid conventional P2PKH address.

This exit should not require:

- a BRC-100 recipient;
- an identity certificate;
- an approved application;
- a wallet-vendor account; or
- continued access to a remote storage provider.

It should be labeled clearly as **Send to Bitcoin Address**, **Transfer to External Wallet** or equivalent. A function available only through developer tools is not meaningful user portability.

Applications should preserve multiple payment paths where practical, including BRC-29/BRC-100 payments, conventional P2PKH outputs, Paymail or compatible payment addressing, and Direct Payment Protocol where appropriate. Users should not become dependent upon one wallet architecture merely to receive or withdraw ordinary BSV.

Recovery requires equally careful treatment. A mnemonic or master key may not be the entire recovery package. Depending on the wallet implementation and assets involved, complete restoration may also require transaction and payment-remittance records—including any required sender identity key, derivation prefix and derivation suffix—along with outpoint records, locking scripts, basket instructions, certificate records and overlay information.

Wallet providers must state plainly whether the seed alone is sufficient. They should never tell users that "the words are the wallet" when additional records are required to find or spend all assets.

Before production deployment, wallet developers should perform a complete recovery test:

1. Fund the original BRC-100 wallet with ordinary BSV and every supported token type.
2. Create transactions across multiple applications and counterparties.
3. Export the full user-controlled backup.
4. Destroy the original installation and local database.
5. Remove the original wallet vendor and remote provider from the recovery path.
6. Restore through an independently installed secondary BRC-100 wallet or open recovery tool.
7. Confirm that all ordinary BSV and supported assets appear.
8. Sweep the ordinary BSV to a conventional legacy address.
9. Transfer each token to an independent compatible implementation without destroying its asset status.

Application developers should separately test whether their application state, tokens and outputs remain usable after the user changes BRC-100 wallets.

A wallet or application that cannot pass the applicable recovery and portability tests should remain experimental and limited to disposable balances.

6. Keep Storage Local and Metadata Minimal

Local storage should be available and preferred by default. Remote storage and synchronization should be optional, separately authorized and clearly explained.

Wallet-Side Storage Duties

When remote storage is used, wallet providers should:

- encrypt sensitive records before transmission wherever technically possible;
- never store the master private key remotely;
- allow the user to select or self-host a provider where practical;
- support multiple backup locations;
- provide a complete export of recovery-critical information;
- publish retention and deletion policies;
- separate authentication logs from financial records;
- minimize IP-address, device and behavioral tracking; and
- demonstrate that recovery remains possible after the provider is removed.

No remote provider should become the sole surviving source of the transaction and derivation records needed to spend the user’s BSV.

Backups should be authenticated, versioned and periodically restored. A backup that has never been tested is only an assumption.

Application-Side Data Duties

Applications should collect and retain only the wallet information needed for their stated function. Access to wallet data should not justify copying that information into a second, indefinitely retained application database.

Applications should disclose whether they retain:

- transaction requests;
- wallet responses;
- payment destinations;
- identity keys;
- certificate fields;
- basket information;
- application-origin records;
- device or analytics identifiers; or
- agent activity logs.

Wallet metadata should also be minimized. Descriptions, labels, tags, basket names and custom instructions should contain only what is operationally necessary. They should not include full legal names, medical information, tax status, political or religious categories, detailed purchase descriptions, investigation flags, unnecessary location data or sensitive account numbers.

Users should be able to export and delete application-specific metadata without losing the underlying ability to spend ordinary BSV.

7. Disable Audit and Linkage Functions by Default

Key-linkage revelation should not be used for ordinary payments, login, token ownership, social posting or reputation.

Applications should not request linkage information merely because the wallet supports it.

Wallets should treat `revealCounterpartyKeyLinkage` and `revealSpecificKeyLinkage` as unusually sensitive operations. They should not be included within ordinary onboarding, persistent background authority or broad grouped permissions.

Where linkage revelation is unavoidable for legal or commercial reasons:

- the narrower specific-key method should be preferred over counterparty-wide revelation;
- the application should explain why it is needed;
- the wallet should require fresh human approval;
- the verifier’s identity and public key should be displayed;
- the exact scope of the revelation should be described;
- the consequences of refusal should be stated;

- background or recurring revelation should be prohibited; and
- the wallet should retain a local record of what was revealed and to whom.

An AI agent should never be permitted to approve linkage revelation, identity-certificate disclosure or expansion of its own authority on behalf of the human owner.

Encryption protects linkage information during transmission. It does not reduce what the approved verifier learns or control what that verifier later does with the information.

8. Preserve Token and Infrastructure Portability

Token portability must be tested separately from ordinary BSV withdrawal.

A token or application asset may depend upon its locking script, basket, custom instructions, transaction ancestry, BEEF data, overlay topic manager, lookup service, issuer and application-specific state. A transaction can remain valid under Bitcoin consensus while an overlay treats the asset as destroyed or no longer recognized.

Application developers should provide:

- a documented external-transfer process;
- export of all required asset records;
- support for more than one compatible wallet where possible;
- avoidance of unnecessary dependence upon one lookup or overlay provider;
- a recovery path if the primary application disappears; and
- clear disclosure where portability does not exist.

Wallet developers should disclose which token, overlay and application-state formats they can preserve during backup, recovery and migration.

An asset should not be marketed as fully user-controlled merely because its UTXO is on-chain when its practical meaning depends upon one proprietary application, index, wallet database or overlay service.

Both applications and wallets should avoid implementation monoculture. Applications should be tested against multiple independently maintained BRC-100 wallets. Wallet providers should avoid exclusive dependence upon one storage provider, authentication backend, certifier, overlay operator or lookup service.

Where common SDK and Wallet Toolbox components are used, developers should pin and review dependency versions, maintain a software bill of materials, monitor security advisories, require code review for wallet-sensitive changes, use automated tests for permissions and recovery, commission independent audits, publish incident procedures and test migration away from the shared component.

An open standard does not eliminate concentration when most implementations rely upon the same underlying code.

9. Make Consent and Disclosure Human-Comprehensible

Permission prompts should describe consequences rather than internal terminology.

Instead of:

> Allow Level 2 access to protocol ID X.

Use:

> Allow this application to sign account-login messages for this service until tomorrow.

Instead of:

> Reveal certificate fields.

Use:

> Send Company X proof that you are over 18. Your name and date of birth will not be sent.

Transaction approvals should display:

- the BSV amount;
- an understandable local-value estimate;
- recipient or requesting application;
- stated purpose;
- whether the payment is one-time or recurring;
- remaining authorization;
- reversibility; and
- whether an AI agent initiated it.

Applications and wallets should not use dark patterns, preselected broad permissions, misleading descriptions or repeated denial loops that pressure users into approval.

10. Publish Separate Application-Side and Wallet-Side Exposure Statements

Transparency should reflect which side of the BRC-100 boundary the product implements.

Application-Side Exposure Statement

Every application connecting to a BRC-100 wallet should disclose:

- which wallet methods it calls;
- which protocol IDs it uses;
- which baskets it requests;
- whether it requests the root identity key;
- which certificates and fields it requests;
- whether it performs identity discovery;
- whether it requests linkage revelation;

- its default spending requests;
- which wallet responses or metadata it retains;
- its supported portability and account-exit methods; and
- whether AI agents can initiate requests through the application.

Wallet-Side Exposure Statement

Every BRC-100 wallet should disclose:

- which BRC-100 methods and capabilities it supports;
- which requests, if any, are permitted automatically;
- which requests require fresh human approval;
- how application origins are authenticated;
- how permissions and data are isolated between applications;
- how identity-key, certificate, basket, spending and linkage requests are presented;
- which permissions persist, how long they last and how they are revoked;
- whether storage is local, remote or both;
- what wallet-request metadata, logs and analytics are retained;
- whether records are shared with storage, authentication or analytics providers;
- what backup information is required for complete recovery;
- whether the wallet can be restored without the original vendor;
- whether ordinary BSV can be sent to a conventional address;
- which token and application assets can be exported or migrated; and
- whether applications or AI agents can act through the wallet and under what hard limits.

A company operating on both sides should publish both disclosures separately.

Any change that expands access, retention, logging, supported identity capabilities or agent authority should require renewed consent and a visible change notice. A broad clause hidden inside a privacy policy is not sufficient.

11. Prepare for Failure Before Launch

Applications and wallets have different failure modes and should identify responsibility before an incident occurs.

Application-Side Failures

Application incident plans should address:

- compromised application origins;
- malicious or altered transaction requests;
- unauthorized expansion of requested permissions;
- application-database breaches;
- misuse of retained wallet metadata;

- compromised AI agents;
- token or overlay failures; and
- disappearance of the application service.

The application should be capable of being revoked without preventing the user from accessing or transferring their assets through the wallet.

Wallet-Side Failures

Wallet incident plans should address:

- origin-authentication failure;
- permission-manager defects;
- improper cross-application access;
- unauthorized signing or spending;
- storage corruption;
- remote-provider outages;
- certificate-verification failures;
- incomplete backup or recovery;
- compromised wallet updates;
- supply-chain vulnerabilities; and
- disappearance of the original wallet implementation.

Emergency controls should allow users to revoke applications, disable agent activity, export wallet state and transfer ordinary BSV to a conventional address without relying upon a compromised application.

Shared Infrastructure Failures

Both sides should prepare for vulnerabilities in shared SDKs, Wallet Toolbox components, authentication providers, storage providers, overlays and certifiers.

Responsibility should not be obscured merely because both products used the same standard or shared dependency.

Developers must describe breaches accurately. Saying “private keys were not exposed” is incomplete when transaction history, application relationships, permission records, certificate information or recovery-critical derivation metadata was compromised.

Conclusion

BRC-100 is versatile, and implementation choices have consequences. The same interface can function as a narrowly restricted payment tool or expand into a common gateway for money, identity, credentials, tokens, applications and autonomous agents.

The safest approach is not to trust the breadth of the standard. It is to resist using that breadth.

Application developers should request only what their products genuinely require. Wallet developers should authenticate, isolate, explain and constrain those requests rather than treating compatibility as automatic authorization.

Developers who proceed should maintain strict boundaries: human savings outside the application wallet; AI agents outside the human wallet; derived application identities instead of universal identity keys; certificates only where necessary; short-lived permissions; local storage by default; independently tested recovery; multiple payment paths; portable token records and a visible legacy exit.

BRC-100 should be treated as a limited operating interface—not as the automatic home for a person’s total wealth, identity and digital life.

Version 2.0 Note: *Thanks to @ProjectBabbage on X for pointing out that the original version did not clearly separate wallet-side and application-side responsibilities. That distinction has been corrected and expanded in this version.*