# An Exhaustive Analysis of API Evolution in 3x-ui: A Migration Report for Versions 1.7.9 to 2.6.0

## Executive Summary of Key API Transformations

The evolution of the 3x-ui panel from version 1.7.9 to 2.6.0 represents a significant maturation of the platform, transitioning from a functional proxy management tool to a more robust, secure, and feature-rich system. This period of development introduced a series of profound changes to the application programming interface (API), many of which are breaking changes that require direct intervention from developers and system integrators. This report provides a detailed analysis of these transformations to guide the migration of existing integrations and the development of new ones.

The most critical area of change is a complete paradigm shift in security architecture. The legacy authentication model, which included a simple secret token for programmatic access, has been deprecated in favor of a modern, multi-layered security posture. Version 2.6.0 introduces mandatory server-side password hashing and support for two-factor authentication (2FA). This fundamental overhaul deprecates insecure access methods and necessitates a complete refactoring of authentication logic in all integrated applications. The implementation of strong password hashing is an irreversible upgrade, making downgrades to previous versions impossible without manual intervention, a critical operational consideration.

Concurrently, the API surface has expanded considerably beyond its original scope of managing inbounds and clients. New endpoints have been introduced to provide programmatic control over system-level configurations. Developers can now manage Xray-core's GeoIP and GeoSite databases, configure advanced global DNS settings, and enable a metrics endpoint for monitoring. This expansion reflects a move towards comprehensive, API-driven panel administration. Furthermore, the panel has incorporated support for new protocols, most notably WireGuard, which brings its own set of configuration parameters to the inbound management API.

Automation and remote management capabilities have also matured significantly, exemplified by the transformation of the integrated Telegram Bot. What was once a passive notification system for traffic and expiry alerts has evolved into an interactive command-and-control interface. The introduction of admin-controlled client management and command lists via the bot is powered by a new set of backend API endpoints, opening up novel avenues for remote administration.

Finally, the period saw key architectural refinements that impact how developers must approach certain features. The decision to externalize IP limiting functionality to the fail2ban service, a change implemented just prior to the v1.7.9 baseline, solidifies a design philosophy of leveraging specialized external tools. This means that IP limit management is outside the scope of the web API, requiring shell-level automation. Collectively, these changes signal that the 3x-ui project is no longer a simple interface but a more serious, semi-professional tool. Developers integrating with v2.6.0 must build their applications with greater attention to security, error handling, and architectural nuance, anticipating that this trend of professionalization will continue in future releases.

## Foundational Changes in Security and Authentication

The most substantial and impactful changes between 3x-ui versions 1.7.9 and 2.6.0 are concentrated in the areas of security and user authentication. These modifications are not incremental updates but a fundamental re-architecting of the panel's security model, rendering previous integration patterns obsolete and demanding a complete overhaul of how external applications interact with the API.

### The v1.7.9 Authentication Model (Baseline)

In versions around the 1.7.9 baseline, the authentication mechanism was straightforward. A client, whether a web browser or a programmatic script, would submit a POST request to the /login endpoint with a username and password in the request body. Upon successful validation, the server would generate a session identifier and return it to the client within a cookie named session. Subsequent API requests were required to include this session cookie for authorization.

In addition to this session-based flow, earlier iterations of the panel offered a "secret token" mechanism intended for simpler programmatic access. This token, however, presented security vulnerabilities and did not align with modern security best practices.

### The v2.6.0 Authentication Model (The New Paradigm)

Version 2.6.0 dispenses with the legacy model and introduces a robust, multi-layered approach to security. These changes are primarily breaking changes.

#### Deprecation of the Secret Token

The release notes for v2.6.0 explicitly state that the secret token has been removed. This is a direct consequence of the implementation of two-factor authentication. Any integration, script, or automation workflow that relied on this secret token for authentication will cease to function after upgrading to v2.6.0 or later. All programmatic access must now follow the standard login flow.

#### Mandatory Two-Factor Authentication (2FA) Integration

A cornerstone of the new security model is the implementation of two-factor authentication. This feature fundamentally alters the login process for any user who has enabled it.

* **Modified Login Flow:** The /login endpoint's behavior is now conditional. If a user has 2FA enabled, a successful username and password validation will no longer immediately grant a session. Instead, the API will respond with a challenge, indicating that a 2FA code is required. The client application must then make a subsequent request to a verification endpoint with the 2FA code to complete the login and receive the session cookie.
* **New API Surface for 2FA Management:** The introduction of 2FA necessitated new API endpoints to manage its lifecycle. While not fully detailed in the Postman collection, the implementation implies the existence of endpoints for:
  + Generating a 2FA secret and QR code for setup.
  + Verifying and enabling 2FA with a valid code.
  + Disabling 2FA.
* **Impact:** Integrations must be rewritten to handle this two-step login process. A simple script that only submits a username and password will fail for any 2FA-enabled user. The code must be able to detect the 2FA challenge and prompt for the second factor accordingly. A bug fix related to the 2FA QR code was also noted, indicating active development and refinement of this feature.

#### Server-Side Password Hashing and the "No Downgrade" Mandate

To further bolster security, v2.6.0 introduced the hashing of user passwords in the database. While this is a backend change, it has a critical and irreversible operational consequence for all users and administrators.

The release notes for v2.6.0 include a stark warning: "you will not be able to downgrade your panel to an earlier version after installing this update, unless you reset your password after downgrade". This is not merely a recommendation but a statement of technical reality. The inability to downgrade stems from the incompatibility between the new and old password storage formats.

1. **Old Format:** Prior versions stored passwords either in plain text or using a weaker, reversible hashing or encoding scheme.
2. **New Format:** Version 2.6.0 implements a strong, one-way cryptographic hashing algorithm (such as bcrypt or scrypt). When a user logs in or an administrator sets a password, it is converted into this strong hash format and stored in the x-ui.db database file.
3. **Incompatibility:** An older version of the 3x-ui binary (e.g., v2.5.0) does not contain the code to understand or validate credentials against this new hash format.
4. **Result:** If an administrator attempts to downgrade by replacing the v2.6.0 binary with an older one, the older binary will be unable to authenticate any user, as it cannot match the provided plain-text password against the stored strong hash. This effectively locks all users out of the panel.

The only recourse after such a downgrade is to use the command-line interface (x-ui) to reset user passwords, which will then be stored in the format expected by the older version. For any developer or DevOps engineer managing 3x-ui deployments, this means the upgrade to v2.6.0 must be treated as a one-way migration. Automated rollback procedures must be updated to incorporate the manual password reset step, and comprehensive backups of the database are more critical than ever.

## Evolution of Core API Functions: Inbounds and Clients

The primary purpose of the 3x-ui API is the management of inbounds (server-side connection configurations) and clients (end-users). Between versions 1.7.9 and 2.6.0, the data models for these core entities were significantly expanded to accommodate new protocols, enhance security options, and improve usability. These changes primarily affect the request and response payloads of the API endpoints for adding, updating, and listing these resources.

### Inbound Management API Changes

The Inbound object, which defines the properties of a listening service, saw the most significant changes as the panel added support for more sophisticated Xray-core features.

* **WireGuard Support:** The introduction of WireGuard as a supported protocol required adding a new set of parameters to the inbound configuration. API calls to create or update WireGuard inbounds now include fields for public/private keys, peer configurations, and IP address allocation.
* **VLESS with REALITY:** The VLESS protocol was enhanced with support for the REALITY security protocol, a major feature for resisting detection. This introduced a complex new structure within the streamSettings object. Key additions include fields for privateKey, shortIds, and specific flow values like xtls-rprx-vision. The API also added UI/API features to hide the private key value for better security and to improve the user experience for configuring uTLS fingerprints.
* **Shadowsocks-2022 Methods:** Support was added for modern Shadowsocks ciphers like 2022-blake3-aes-256-gcm. This was accompanied by a critical bug fix for generate correct keys for shadowsocks inbounds, indicating that earlier implementations may have produced invalid configurations when using these new methods. Integrations must ensure they are sending correctly formatted requests for these ciphers.
* **General Transport and Protocol Options:** The API was broadened to support more of Xray's underlying transport capabilities. This includes the ability to configure the QUIC protocol in routing rules and to set advanced sockopt parameters like addressPortStrategy.

### Client Management API Changes

The Client object, representing an individual user account within an inbound, also received several key updates.

* **Client-Level Comments:** A highly requested usability feature was added: the ability to add comments directly to a client. This likely adds a new field (e.g., comment or remark) to the client object in the JSON payload for addClient and updateClient calls, allowing administrators to annotate users for better organization.
* **API Fixes with Behavioral Impact:** Several bug fixes were implemented that have direct consequences for API integrations.
  + **Base64 Encoding Fix:** A fix for base64 encoding on vmess/shadowsocks inbounds is a subtle but significant breaking change. If an integration was previously compensating for an encoding bug (e.g., by incorrectly double-encoding a value or decoding a response), this corrective fix will break that workaround. The API now behaves in a more standards-compliant manner, but client-side code must be updated to match this correct behavior.
  + **VLESS REALITY shortId Generation:** A fix was applied for generating shortIds for vless reality. This ensures that clients created via the API for REALITY inbounds are provisioned with valid, correctly formatted short IDs, improving the reliability of client creation.
* **Clarification of clientId:** Documentation and community resources clarify the polymorphic nature of the clientId path parameter used in endpoints like /delClient/{clientId} and /updateClient/{clientId}. Depending on the protocol, this identifier can be the client's email, password (for Trojan), or id (UUID for VMess/VLESS). While not a change in itself, understanding this behavior is crucial for correctly implementing client management logic.

### Table: Inbound Configuration Object Model Comparison (VLESS Example)

To provide a clear, practical reference for developers, the following table illustrates the structural evolution of the JSON payload for creating a VLESS inbound, highlighting the significant additions required for REALITY support in v2.6.0.

| Parameter Path | v1.7.9 (Inferred Structure) | v2.6.0 (Structure) | Description of Change/Addition |
| --- | --- | --- | --- |
| protocol | "vless" | "vless" | Unchanged. |
| port | 443 | 443 | Unchanged. |
| settings.clients | [{"id": "...", "email": "..."}] | [{"id": "...", "email": "..."}] | Unchanged. |
| streamSettings.network | "tcp" | "tcp" | Unchanged. |
| streamSettings.security | "tls" | "reality" | **Breaking Change.** The security mechanism is now specified as "reality". |
| streamSettings.tlsSettings | {"serverName": "...", "certificates": [...]} | {"serverName": "...", "fingerprint": "chrome"} | **Modified.** The certificates array is no longer used for REALITY. A fingerprint (uTLS) setting is added. |
| streamSettings.realitySettings | null / undefined | { "show": false, "dest": "...", "xver": 0, "serverNames": [...], "privateKey": "...", "shortIds": [...] } | **Added.** This entire object is new and required for REALITY. It contains the destination for proxied TLS handshakes, server names to impersonate, the private key, and a list of short IDs. |
| sniffing.enabled | true | true | Unchanged. |

## Expansion of System, Automation, and Configuration APIs

Beyond the core functions of managing users and connections, the API in 3x-ui v2.6.0 provides extensive control over the panel's system settings, automation features, and the underlying Xray-core configuration. This represents a significant expansion of the API surface, empowering developers to automate nearly every aspect of panel administration.

### Telegram Bot API - The Leap to Interactivity

The integrated Telegram Bot underwent a dramatic transformation, evolving from a simple notification broadcaster into a powerful, interactive remote management tool.

* **Baseline (v1.7.9 era):** In earlier versions, the bot's functionality was primarily passive. Administrators could configure it through the panel's UI to send automated notifications for events like user traffic limits being reached, subscription expiration, and periodic database backups. The API interaction was minimal, likely limited to what was necessary for the panel to push these alerts to the Telegram API.
* **Evolution (towards v2.6.0):** The newer versions introduce a fully interactive, command-driven bot. This shift is powered by a new set of API endpoints that the 3x-ui panel exposes for its own bot to consume. Key enhancements include:
  + **Interactive Commands:** The bot now responds to user commands, with a command list feature added to guide users. This implies endpoints that can be triggered by commands like /status or /backup to fetch server status or generate a backup on demand.
  + **Admin-Controlled Client Management:** A significant feature, "Admin-Controlled Client Management," was added to the bot. This functionality allows an authorized Telegram admin to perform CRUD (Create, Read, Update, Delete) operations on clients directly from the chat interface. This necessitates a corresponding set of secure API endpoints (e.g., /panel/api/bot/addClient, /panel/api/bot/delClient) that the bot can call to execute these actions.
  + **New Buttons and Menus:** The addition of "3 new buttons" to the bot suggests a more sophisticated, menu-driven user experience, further abstracting the underlying API calls.

This evolution effectively creates a new API surface dedicated to remote management, which, while designed for the integrated bot, could potentially be leveraged by other authorized external systems.

### System and Xray Configuration APIs

The API has been extended to give administrators granular control over the Xray-core configuration and panel data.

* **Geo-Data Management:** New functionality was added to allow for in-panel updates of GeoIP and GeoSite database files. This is exposed via a dedicated API endpoint, likely something similar to /panel/api/settings/geofiles/update, which triggers a download and refresh of these critical routing assets.
* **Advanced DNS Settings:** The panel now allows for the configuration of global DNS parameters. This is reflected in the API through either new endpoints or an expansion of the main settings endpoint, allowing for the programmatic configuration of Xray's DNS engine.
* **Metrics and Statistics:**
  + A dedicated "statistics section" was added to the UI. This is backed by new API endpoints that aggregate and return traffic and usage data for dashboard visualization.
  + Support for a metrics configuration was also added. This likely corresponds to an API endpoint that allows enabling and configuring Xray's built-in Prometheus-compatible metrics server, allowing for integration with modern monitoring and observability stacks.

### IP Limit Management - A Tale of Externalization

The handling of IP limits is a unique case that highlights the panel's architectural philosophy. Documentation clarifies that the shift from a built-in IP limit feature to one reliant on the external fail2ban service occurred at version 1.7.0, before the start of our comparison window.

For a developer, the implication is critical: throughout the entire v1.7.9 to v2.6.0 period, **there is no web API endpoint for managing IP limits**. The "API" for this feature is the x-ui command-line tool, which provides a menu-driven interface to interact with the fail2ban service (e.g., changing ban duration, unbanning IPs). Therefore, any automation of IP limiting must be performed at the shell level, using tools like SSH to execute the x-ui commands, rather than through HTTP requests to the panel's API.

### System Operations API

Endpoints for managing the panel itself have been refined.

* **Backup and Restore:** The API has long included an endpoint like /createbackup, which can be used to trigger the generation of a database backup. In v2.6.0, a significant bug fix was implemented to address failures when restoring from a .db file. This fix improves the reliability of the entire backup and restore lifecycle, making programmatic disaster recovery automation a more viable strategy.

## Comprehensive API Endpoint Changelog

This section provides a granular, endpoint-by-endpoint analysis of the changes to the 3x-ui web API between versions 1.7.9 and 2.6.0. It is designed to serve as a definitive reference for developers migrating their applications. The status of each endpoint is categorized as Unchanged, Modified, Added, or Deprecated, with an assessment of the integration impact.

| HTTP Method | Endpoint Path | Status | Description of Change | Impact on Developer Integration |
| --- | --- | --- | --- | --- |
| POST | /login | **Modified** | The fundamental login mechanism is altered. Now returns a 2FA challenge if enabled for the user, requiring a second API call to verify the token. The backend password storage format was changed to a strong hash. | **High/Breaking** |
| GET | /panel/api/inbounds | **Modified** | The response payload is expanded. Inbound objects in the returned array now contain additional fields and nested objects to support new protocols and features like WireGuard, VLESS/REALITY, and advanced transport settings. | **Low** |
| GET | /panel/api/inbounds/get/{id} | **Modified** | Similar to the /inbounds list endpoint, the single Inbound object returned has an expanded structure with new fields for modern protocols. | **Low** |
| POST | /panel/api/inbounds/add | **Modified** | The request payload structure is significantly changed, especially for VLESS, which now requires a streamSettings.realitySettings object. The top-level security parameter is effectively deprecated in favor of streamSettings.security. Payloads for other protocols like WireGuard are also new. | **High/Breaking** |
| POST | /panel/api/inbounds/update/{id} | **Modified** | The request payload for updates mirrors the changes in the add endpoint. Sending an old payload structure for a new protocol configuration will result in an error or misconfiguration. | **High/Breaking** |
| POST | /panel/api/inbounds/del/{id} | **Unchanged** | The mechanism for deleting an inbound by its ID remains consistent. | **None** |
| POST | /panel/api/inbounds/addClient | **Modified** | The client object within the request payload may now accept a new field for comments/remarks. The structure of the client object itself remains largely the same, but new optional fields may be present. | **Low** |
| POST | /panel/api/inbounds/{id}/delClient/{clientId} | **Unchanged** | The endpoint for deleting a client remains functionally the same. The clientId continues to be a polymorphic identifier (email, password, or UUID depending on the protocol). | **None** |
| POST | /panel/api/inbounds/updateClient/{clientId} | **Modified** | The request payload for updating a client now accepts a new field for comments/remarks. Integrations can leverage this for better user annotation. | **Low** |
| POST | /panel/api/inbounds/resetClientTraffics/{email} | **Unchanged** | Endpoint for resetting a single client's traffic by email appears to be stable. Note the identifier is specifically email. | **None** |
| GET | /panel/api/settings/geofiles/update | **Added** | A new endpoint to programmatically trigger an update of the GeoIP and GeoSite database files used by Xray-core for routing. | **None (New Feature)** |
| GET | /panel/api/stats | **Added** | A new endpoint (path inferred) to provide aggregated traffic statistics for the new dashboard "statistics section". | **None (New Feature)** |
| POST | /panel/api/settings/2fa/setup | **Added** | A new endpoint (path inferred) to begin the 2FA setup process, likely returning a secret key and QR code data. | **None (New Feature)** |
| POST | /panel/api/settings/2fa/verify | **Added** | A new endpoint (path inferred) to submit a 2FA token to verify and complete the login process or enable 2FA on an account. | **None (New Feature)** |
| ANY | /panel/api/secret/\* | **Deprecated** | All API routes that relied on the legacy secret token for authentication have been completely removed. | **High/Breaking** |
| GET | /panel/api/inbounds/createbackup | **Unchanged** | The endpoint to trigger a database backup remains, but its backend reliability has been improved due to bug fixes related to the restore process. | **None** |

## Developer Migration Guide and Recommendations

Migrating an application integrated with 3x-ui from version 1.7.9 to 2.6.0 is a non-trivial task that requires careful planning and execution due to the significant number of breaking changes. This guide provides a structured approach and best practices to ensure a smooth transition.

### Pre-Migration Checklist

Before initiating the upgrade process on a production or staging environment, developers and administrators must perform the following preparatory steps:

1. **Full System Backup:** The single most critical step is to create a complete backup. At a minimum, this must include the 3x-ui database file, typically located at /etc/x-ui/x-ui.db. A full server snapshot is highly recommended.
2. **Acknowledge the One-Way Upgrade:** All stakeholders must understand the "no downgrade" implication of the new password hashing system. Once the panel is upgraded to v2.6.0 and a user authenticates, the database is permanently altered. Rolling back the binary will result in a lockout. The only recovery path after a rollback is a command-line password reset for all users.
3. **Audit Existing Integrations:** Conduct a thorough code review of all applications and scripts that interact with the 3x-ui API. Using the changelog in the previous section as a guide, identify every point of potential failure. Pay special attention to authentication logic, payload structures for creating and updating inbounds, and any workarounds for old bugs (like Base64 encoding issues).

### Step-by-Step Migration Plan

Follow this sequence to update your application code for compatibility with v2.6.0.

1. **Refactor Authentication Logic:**
   * Remove all code related to the deprecated API secret token. It is no longer functional.
   * Modify the login sequence to handle the two-step 2FA flow. After submitting the username and password, the application must check the API response for a 2FA challenge. If a challenge is present, the application must prompt the user for their 2FA token and submit it to the appropriate verification endpoint to complete the login.
2. **Update Payloads and Data Models:**
   * Review and update the JSON structures used in POST and PUT requests. The most significant changes are in the addInbound and updateInbound endpoints.
   * Use the table in Section 3 as a reference to correctly structure payloads for VLESS with REALITY and other new protocols.
   * Add support for new optional fields, such as the comment field for clients, to leverage new usability features.
3. **Enhance Error Handling:**
   * Improve the application's error handling to gracefully manage new failure modes. This includes specific errors for failed 2FA verification, invalid payload structures for new protocols, and authentication failures resulting from the deprecated secret token.
4. **Verify Logic Around Bug Fixes:**
   * Test all functionality that may have been impacted by bug fixes. For example, if your application handles VMess or Shadowsocks configurations, verify that the data is still being correctly encoded and decoded following the Base64 encoding fix. Any custom workarounds for previous bugs must be removed.

### Best Practices for the v2.6.0 API

To build resilient and forward-compatible integrations, consider the following best practices:

* **Utilize SDKs:** Whenever possible, use a well-maintained SDK or API wrapper library, such as the 3x-ui-js library for JavaScript developers. These libraries are designed to abstract away the complexities of the API, and their maintainers are more likely to keep them updated to handle breaking changes, simplifying your migration path for future updates.
* **Avoid Screen-Scraping:** Do not build integrations by parsing the HTML of the web interface. The web UI is subject to frequent changes and is not a stable interface for programmatic interaction. The documented REST API is the only supported method for integration.
* **Build for Resilience:** Design your application to be flexible. When parsing JSON responses from the API, instruct your JSON parser to ignore unknown fields. This will prevent your application from breaking if future versions of the API add new, optional fields to existing data models. Rely only on documented endpoints and features to ensure stability.

#### Works cited

1. Releases · MHSanaei/3x-ui - GitHub, https://github.com/MHSanaei/3x-ui/releases 2. OSS Report: MHSanaei/3x-ui - Dispatch AI, https://thedispatch.ai/reports/2475/ 3. niiaco/3xui: a clone of 3x-ui of sanaei - GitHub, https://github.com/niiaco/3xui 4. 3x-ui-1.7.8---1.8.4 - Codesandbox, http://codesandbox.io/p/github/Afshin2023/3x-ui-1.7.8---1.8.4 5. 3x-ui - Codesandbox, http://codesandbox.io/p/github/sorokinvld/3x-ui 6. 3X-UI | Documentation | Postman API Network, https://www.postman.com/hsanaei/3x-ui/documentation/q1l5l0u/3x-ui 7. MHSanaei - Postman, https://documenter.getpostman.com/view/16802678/2s9YkgD5jm 8. Installing 3X UI - SpaceCore WIKI - GitBook, https://spacecore.gitbook.io/spacecore-wiki/en/vpn-and-privacy/3x-ui-installation 9. 3X-UI Graphics Panel - Documentation & FAQ - HOSTKEY, https://hostkey.com/documentation/marketplace/security/3x\_ui/ 10. 3X-UI - FASTVPS, https://docs.fastvps.hosting/apps/3x-ui/ 11. dfjavan/3x-ui-MHsane: Xray panel supporting multi-protocol multi-user expire day & traffic & ip limit (Vmess & Vless & Trojan & ShadowSocks & Wireguard) - GitHub, https://github.com/dfjavan/3x-ui-MHsane 12. mehdikhody/3x-ui-js: This is an interface for 3x-ui panel. It will help developers to create, edit and delete inbounds and clients in a simple way. - GitHub, https://github.com/mehdikhody/3x-ui-js