Using the Native Jetpack Connection Flow in PR #22782

PR: #22782 — CMM-1949: Fix Application Password login when XML-RPC is unavailable
Date: 2026-04-09

Background

PR #22782 adds a new `XmlRpcDisabledBottomSheetFragment` with a "Connect Jetpack" button. Currently, this button launches the **web-based** Jetpack connection flow via `JetpackConnectionWebViewActivity`, which opens a WebView to `wordpress.com/jetpack/connect`.

The app also has a **native REST-based** Jetpack connection flow (`JetpackRestConnectionActivity`) that handles the entire connection through direct API calls with a native UI. This document describes how the bottom sheet could use the native flow instead.

Current Implementation

In `XmlRpcDisabledBottomSheetFragment.kt` (lines 47–56):

```
binding.connectJetpackButton.setOnClickListener {
    JetpackConnectionWebViewActivity
        .startJetpackConnectionFlow(
            requireActivity(),
            JetpackConnectionSource.XMLRPC_DISABLED,
            site,
            accountStore.hasAccessToken()
        )
    dismiss()
}
```

This always uses the web-based flow regardless of whether the native flow is available.

Native Flow Overview

The native flow lives in `JetpackRestConnectionActivity` and `JetpackRestConnectionViewModel`. It performs a 5-step process entirely through API calls (using wordpress-rs):

1. **LoginWpCom** — Verify or login to WordPress.com

2. **InstallJetpack** — Install the Jetpack plugin via REST API
3. **ConnectSite** — Connect the site to Jetpack
4. **ConnectUser** — Connect the user to Jetpack
5. **Finalize** — Activate the stats module

Eligibility Requirements

The native flow has strict eligibility requirements, checked by `canInitiateJetpackRestConnection(site)`:

```
fun canInitiateJetpackRestConnection(site: SiteModel): Boolean {
    return BuildConfig.IS_JETPACK_APP
            && site.isUsingSelfHostedRestApi
            && !site.wpApiRestUrl.isNullOrEmpty()
            && !site.isJetpackConnected
            && (!site.isJetpackInstalled
                || checkMinimalVersion(
                    site.jetpackVersion, "14.2"))
}
```

Condition	Reason
`IS_JETPACK_APP`	Only available in the Jetpack app, not the WordPress app
`isUsingSelfHostedRestApi`	Site must use the self-hosted REST API
`wpApiRestUrl` not empty	REST API URL must be configured
Not Jetpack-connected	No point connecting if already connected
Jetpack not installed or ≥ 14.2	Older Jetpack versions don't support the native flow

> **Note:** Sites logged in via Application Password with XML-RPC disabled will have `isUsingSelfHostedRestApi = true` and `wpApiRestUrl` set, so they should pass the eligibility check when running in the Jetpack app.

Existing Fallback Pattern

Both `NotificationsListFragment` and `StatsConnectJetpackActivity` use the same pattern to prefer the native flow with a web fallback:

```
// From NotificationsListFragment.kt
private fun startJetpackRestConnectionFlow(
    site: SiteModel
```

```
): Boolean {
    if (JetpackRestConnectionViewModel
            .canInitiateJetpackRestConnection(site)) {
        JetpackRestConnectionActivity
            .startJetpackRestConnectionFlow(
                requireActivity(),
                ConnectionSource.NOTIFS
            )
        return true
    }
    return false
}

// Usage:
if (!startJetpackRestConnectionFlow(selectedSite)) {
    // Fallback to web-based flow
    JetpackConnectionWebViewActivity
        .startJetpackConnectionFlow(
            activity, NOTIFICATIONS, selectedSite, false
        )
}
```

Required Changes

Step 1: Add `XMLRPC_DISABLED` to `ConnectionSource`

File: `JetpackRestConnectionViewModel.kt`

```
enum class ConnectionSource {
    STATS,
    NOTIFS,
    XMLRPC_DISABLED  // ← New value
}
```

Step 2: Update the bottom sheet button handler

File: `XmlRpcDisabledBottomSheetFragment.kt`

```
binding.connectJetpackButton.setOnClickListener {
    if (!startJetpackRestConnectionFlow(site)) {
        // Fallback to web-based flow
        JetpackConnectionWebViewActivity
            .startJetpackConnectionFlow(
                requireActivity(),
                JetpackConnectionSource.XMLRPC_DISABLED,
                site,
                accountStore.hasAccessToken()
            )
    }
```

```
    dismiss()
}

private fun startJetpackRestConnectionFlow(
    site: SiteModel
): Boolean {
    if (JetpackRestConnectionViewModel
            .canInitiateJetpackRestConnection(site)) {
        JetpackRestConnectionActivity
            .startJetpackRestConnectionFlow(
                requireActivity(),
                JetpackRestConnectionViewModel
                    .ConnectionSource.XMLRPC_DISABLED
            )
        return true
    }
    return false
}
```

Step 3: Add imports

File: `XmlRpcDisabledBottomSheetFragment.kt`

```
import
org.wordpress.android.ui.jetpackrestconnection.JetpackRestConnectionActivity
import
org.wordpress.android.ui.jetpackrestconnection.JetpackRestConnectionViewModel
```

No new Dagger injections are required — the eligibility check is a static method and the activity is launched via a companion object method.

Behavior Summary

Scenario	Flow Used
Jetpack app + REST API + not connected	Native (5-step API flow)
WordPress app (any configuration)	Web-based (WebView fallback)
Jetpack app + already connected	Web-based (WebView fallback)
Jetpack app + old Jetpack version (< 14.2)	Web-based (WebView fallback)

Files Modified

File	Change

`JetpackRestConnectionViewModel.kt`	Add `XMLRPC_DISABLED` to `ConnectionSource` enum
`XmlRpcDisabledBottomSheetFragment.kt`	Add eligibility check + fallback pattern, new imports