Jenkins Email Notifications Using Outlook SMTP with OAuth

Google Summer of Code Program 2026 Project Proposal

Name: Shri Raj Bisaria

Email: lucid7006@gmail.com

GitHub: subwaycookiecrunch

Project Abstract

The email-ext plugin has an OAuth2 checkbox that sets the XOAUTH2 JavaMail flag, but nothing behind it actually acquires a token , it still sends the plain password. This project adds the missing pieces: a credential type for Azure AD app registrations, token acquisition via Microsoft's client credentials grant, caching, and the UI work to make it usable. Once done, Jenkins admins can send build notifications through Outlook SMTP after Microsoft finishes killing basic auth.

Project Description

Microsoft has been deprecating basic authentication for Exchange Online. When that process completes, Jenkins instances using email-ext with username/password SMTP auth to Outlook will stop being able to send email. There's no workaround in the plugin right now.

email-ext is one of the most installed Jenkins plugins , over 100k installations. A lot of those are corporate setups running Office 365. This is going to break things for a lot of people.

I went through the plugin source before writing this. Here's what the codebase actually looks like right now:

`MailAccount.java` stores a `useOAuth2` boolean. Checking the box in the UI sets this flag. But the form validator `doCheckCredentialsId` didn't even receive this flag as a parameter, so it couldn't validate anything OAuth-related. I fixed that in PR #1467 , the validator can now warn about TLS when OAuth2 is enabled.

`ExtendedEmailPublisherDescriptor.java` creates the JavaMail session. Around line 317, `createSession()` sets `mail.smtp.auth.mechanisms=XOAUTH2` when the flag is on. That part works. The problem is `getAuthenticator()` at line 393 , it always creates a plain `PasswordAuthentication` with whatever's in the credentials store. Doesn't matter if OAuth2 is on or off, same code path. For

XOAUTH2 to work, that password field needs to contain an actual access token from Microsoft, not a user's password.

config.groovy shows the OAuth2 checkbox under "Advanced" but gives you no way to enter a tenant ID, client ID, or scope. Just the checkbox and the regular username/password credential dropdown.

So the flag exists, JavaMail gets told to use XOAUTH2, but nobody actually talks to Microsoft to get a token. That's the gap.

How I'd build it

Credential type. OAuth2 client credentials need a tenant ID, client ID, and client secret. StandardUsernamePasswordCredentials doesn't fit , wrong shape entirely. I'd add a new credential type in the plugin implementing StandardCredentials, with its own Descriptor and config view.

I considered depending on the existing Microsoft Entra OAuth plugin instead. It would save some work, but I'm not sure it exposes a credential type with the right fields, and adding a hard dependency on another plugin isn't great. I want to talk this through with the mentors before committing either way.

The scope defaults to https://outlook.office365.com/.default , that's what Microsoft requires for SMTP.

Token acquisition. The client credentials grant:

```
POST https://login.microsoftonline.com/{tenant}/oauth2/v2.0/token
Content-Type: application/x-www-form-urlencoded

client_id={client_id}
&scope=https://outlook.office365.com/.default
&client_secret={client_secret}
&grant_type=client_credentials
```

Response is JSON with access_token and expires_in (usually 3599 seconds). I'd write an OAuthTokenProvider class using java.net.http.HttpClient , no extra dependency needed since Jenkins requires Java 17+.

Caching. Tokens are good for about an hour. Hitting the token endpoint on every build email is wasteful and could hit rate limits on busy instances. I'd cache per MailAccount and refresh when the token is within 5 minutes of expiring. Just the token string and an expiry timestamp in memory. If Jenkins restarts, it fetches a new one on the next email send.

The actual wiring. getAuthenticator() needs a branch: when acc.isUseOAuth2() is true, look up the OAuth2 credential instead of the username/password one, get a (cached) token from OAuthTokenProvider, and return PasswordAuthentication(email, token). The token goes in the password field because that's how XOAUTH2 works with JavaMail.

UI. When the OAuth2 checkbox is on, the credential dropdown should filter to only show OAuth2 credentials. I'd add a "Test Connection" button that tries to acquire a token and tells you if it worked. And some help text about Azure AD app registration , that process is confusing if you haven't done it.

Testing. Unit tests for the token provider with mocked HTTP (successful fetch, expired token, error responses). Integration test for the authenticator. JCasC test to make sure OAuth2 config survives a YAML export/import cycle.

Why I picked this project

I was using Jenkins for a university project and dealt with the email notification setup firsthand. When I saw this on the GSoC ideas list, I looked at the code and realized the OAuth2 checkbox didn't actually do anything. It's a clear, well-scoped problem and I'd already been poking around the email-ext codebase.

Project Deliverables

Dates from the GSoC Timeline take precedence.

- **Community Bonding (May 1 – May 24):** Dev environment working, can build and test the plugin locally. Credential type approach decided with mentor input. Manual OAuth flow tested with curl against a real Azure AD tenant.

- **Midterm Evaluations (July 6 – July 10):** Credential type done, token provider working with caching, authenticator wired up. Should be able to actually send an email through Outlook using OAuth at this point. Draft PR up for review.

- **Final Evaluations (August 17 – August 24):** UI polish (credential filtering, test button, help text). JCasC support. Improved validation. Error handling for revoked apps, network issues, FIPS. Documentation. Final PR ready.

Proposed Schedule

12-15 hours per week. I'll push code almost every day during coding phases.

March 16 – March 31 (Application Period)

Keep contributing to Jenkins and email-ext. Respond to any reviewer feedback on open PRs. If mentors engage on Discourse, start discussing the credential type question early.

April 1 – April 30 (Acceptance Waiting Period)

More contributions regardless of outcome. Read deeper into the Microsoft identity platform docs — especially around multi-tenant vs single-tenant apps and token revocation behavior. Set up a test Azure AD app registration.

May 1 – May 24 (Community Bonding)

Get to know the mentors. Make sure I can not just build the plugin but also run its full test suite and deploy a local Jenkins with it loaded. Test the OAuth flow manually with curl so I understand exactly what the requests look like. Nail down the credential type approach.

May 25 – July 6 (Coding Period 1)

First two weeks: the credential type. New class implementing StandardCredentials, Jelly config views, tests for storage and retrieval.

Next two weeks: OAuthTokenProvider and the caching layer. Unit tests with mocked HTTP for all the different responses Microsoft can send back.

Last two weeks: wire it into getAuthenticator(), get end-to-end working. Actually send an email through Outlook with OAuth. Put up a draft PR.

July 6 – August 24 (Coding Period 2)

Weeks 7-8: UI work. Credential dropdown filtering, Test Connection button, help text for the Azure AD setup.

Weeks 9-10: JCasC support. Form validation improvements , right now you can check OAuth2 and save without any OAuth credentials configured, which silently breaks email sending.

Weeks 11-12: Edge cases. Revoked app registrations (clear cached token, surface a useful error). Network failures during refresh. FIPS mode compatibility , the plugin already has MailAccountFIPSTest.java so I have a reference for how to handle this. Documentation and cleanup.

Post-GSoC

I want to stick around. OAuth support will need maintenance as Microsoft's platform changes, and I'll know the code best. There are also other email-ext issues I'd like to work on.

Future Improvements

Things that could follow the core OAuth work:

- **Gmail OAuth.** Google also requires OAuth for SMTP now. Different token endpoint and scope, but the plugin architecture could be extended to handle multiple providers.

- **Delegated auth.** Client credentials sends email "as the application." Some orgs need it to come from a specific user's mailbox, which means the authorization code flow. More complex but worth exploring.

- **Token failure monitoring.** If a token stops refreshing (admin revoked the app or the secret expired), surfacing that as a Jenkins admin monitor instead of just a failed email build step.

Continued Involvement

I'd like to keep maintaining the OAuth code after GSoC. Microsoft changes things periodically and someone who knows the implementation should be around to handle that. Beyond OAuth, there are open issues around email templating and how attachments work that I'd want to look at. If the maintainers are open to it, I'd be interested in taking on more responsibility with the plugin over time.

Conflict of Interests or Commitments

Full-time university student. No summer internship, job, or fellowship that would conflict.

- **University classes:** Classes run Monday–Tuesday 8 AM – 5 PM and Wednesday–Friday 12 PM – 5 PM (IST).

- **Exams:** Mid April

- **Other commitments:** None during the program.

- **Hours per week:** 20-25, more during breaks.

- **Code ownership:** My university doesn't claim ownership of code written outside coursework. No restrictions on contributing.

Major Challenges Foreseen

The credential type question is the biggest design decision. Making a new type keeps things self-contained but means more code to maintain. Depending on an existing OAuth plugin is less work but creates coupling and I'm not sure it has everything we need. This is why I want to resolve it during community bonding, not while coding under a deadline.

End-to-end testing is tricky because you need a real Azure AD app registration. Unit tests use mocks, but actually confirming it works requires hitting Microsoft's token endpoint. I'll set this up during community bonding so it's ready before coding starts.

FIPS mode restricts certain crypto operations in Jenkins. I need to check that the HTTP client used for token acquisition doesn't violate those constraints. Haven't looked deeply into this yet but the existing MailAccountFIPSTest.java should tell me what to watch for.

Backwards compatibility. Everything I'm adding needs to be behind the OAuth2 flag. If you don't enable it, the plugin should behave exactly like it does today. No surprises for existing users.

References

- Microsoft OAuth 2.0 Client Credentials Grant
- Authenticating SMTP with OAuth (Microsoft docs)
- email-ext-plugin source
- Jenkins Credentials API consumer guide
- JCasC plugin
- JavaMail / Angus Mail
- GSoC project idea page

Relevant Background Experience

Jenkins Core — 4 PRs

PR #26361 — "Manage Old Data" page was re-resolving class names on every load. Added a ConcurrentHashMap cache. Had to understand RobustReflectionConverter internals to get the invalidation right. Most involved fix I've done so far.

PR #26364 — "Keep this build forever" button was missing on the new build page. The new OverviewTab/index.jelly didn't include logKeep.jelly. Tiny fix but took a while to trace through the Jelly templates.

PR #26354 — Java Records crashed XStream2 deserialization.

PR #26353 — Agent ping interval config.

email-ext-plugin — 2 PRs

PR #1467 — doCheckCredentialsId() wasn't receiving the useOAuth2 flag, so it couldn't validate OAuth settings. Added the parameter.

PR #1468 — Attachments with a Content-ID were missing Content-Disposition: inline, so they showed up as file downloads in Teams instead of inline images.

contributor-spotlight — 1 PR

PR #528 — Placeholder image fallback when profile pictures fail to load.

Personal

Name: Shri Raj Bisaria

Email: lucid7006@gmail.com

University / School: Galgotias University

Level: 2nd year B.Tech Computer Science

About Me

I'm a second-year CS student from India, graduating 2028. I work mostly in Java and got into Jenkins while browsing GSoC project ideas , the email OAuth project stood out because the problem was concrete and I could see exactly what needed to change in the code. I've been contributing to Jenkins core and the email-ext plugin since then. Outside of coursework I tinker with blockchain and decentralized systems.

How did you learn about Jenkins?

I'd heard of Jenkins in university (CI/CD courses / DevOps discussions / whatever's true), but I started actually reading the code when I saw it on the GSoC 2026 ideas list.

Availability and Commitments

- **University schedule:** Classes run Monday–Tuesday 8 AM – 5 PM and Wednesday–Friday 12 PM – 5 PM (IST).
- **Exams:** 15th April 2026
- **Other commitments:** None.
- **Hours:** 20-25/week, more during breaks.
- **Weeks unavailable:** None expected

Experience

Free Software Contributions

7 PRs to Jenkins core, email-ext-plugin, and contributor-spotlight (listed above).

[ADD ANY OTHER OPEN SOURCE WORK]

Language Skill Set

Language	Level
Java	Intermediate — worked with Jenkins plugin model, Maven, JavaMail
JavaScript	Intermediate — React/Gatsby (from contributor-spotlight work)
Python	Intermediate — scripting and general-purpose use
HTML/CSS	Comfortable — can build and style pages
Rust	Familiar — contributed to open source Rust projects
C++	Intermediate — competitive programming

Related Research and Work Experience

No formal industry experience yet. University projects and personal work include:

- **Lockless Concurrent Framework** — A C++20 framework with lock-free data structures (ring buffer, stack, hash map) and profiling tools for low-latency systems. Dealt with cache alignment, memory ordering, and correctness validation.
- **TTBD — Time-Traveling Blockchain Debugger** — A Rust tool for stepping through blockchain state transitions. Wrote the core module structure and state management.
- Contributed to **Foundry** (Ethereum development toolkit, Rust), **RustDesk** (remote desktop, Rust), and **Apache DataFusion** (SQL query engine, Rust).

Reference Links

- GitHub: https://github.com/subwaycookiecrunch
- Linkedin: www.linkedin.com/in/shri-raj-bisaria-259a38382