# Why You Should Not Mix Terraform and Akamai API Scripts

Mixing Terraform scripts and direct Akamai API call scripts (e.g., Python with EdgeGrid) in a single solution is generally not recommended. Below is a detailed explanation outlining the risks and best practices.

## Reasons Not to Mix Terraform and Akamai API Scripts

### 1. State Drift and Conflict

Terraform maintains its own state file and expects full ownership of all managed resources. API scripts do not update this state, which can cause drift between what Terraform believes exists and what is actually deployed. This may lead to unexpected deletions, recreations, or errors.

### 2. Loss of Idempotency

Terraform ensures that the infrastructure matches the code consistently. API scripts, being imperative, can make changes that are not reflected in the Terraform state, which breaks Terraform’s idempotent behavior.

### 3. Ownership Ambiguity

When both Terraform and API scripts manage the same resources, it’s unclear which tool has authority. Terraform might overwrite or destroy changes made by API scripts, leading to operational issues.

### 4. Difficult Troubleshooting

When things go wrong, it’s hard to identify the source of the change. Mixing tools increases the complexity of debugging and root cause analysis.

### 5. Auditability and Compliance Risks

Terraform provides version control and logging for compliance. API scripts usually don’t, which can lead to gaps in audit trails and violate governance policies.

### 6. Module and Dependency Breakage

Terraform modules assume they control their resources. API script modifications inside those modules can break dependencies and cause Terraform operations to fail.

## Summary

|  |  |
| --- | --- |
| Risk of Mixing | Impact |
| State Drift | Inconsistent resource behavior |
| Terraform apply breaks | Unwanted deletion or updates |
| Hard to audit | Poor compliance visibility |
| Resource duplication or overwrite | Unpredictable and hard to debug |