Change Management Policy

Version 1 - Approved by Abdullah Adeeb

Change Management Policy

This policy outlines the principles we apply to handle changes we intend to make in IT operational environments. Our aim is to enable these changes safely, effectively, and efficiently. In other words, these guidelines help us reduce the risk of experiencing platform or application instability and increase the predictability and reproducibility of the change management process.

Scope

This policy covers all systems within our environment. The environment includes all endpoints and cloud assets used in hosting and its subdomains. This doesn't include third-party systems that support the business of AutoRx Solutions.

Trust

Our change management is underpinned by trust: we trust ourselves to act responsibly when working in the operational environment and do everything in our power to safeguard its integrity and maintain its availability and performance.

To that end, we must develop a practice that allows for safe changes. Our highest priority is always the integrity and reliability of the operational environment, which entails appropriate risk evaluation, quantifiable validation and verification, extensive communication, detailed auditing, and a focus on defensive coding.

All software is version controlled

All software developed in the service of AutoRx Solutions's offering should be version-controlled i.e the latest version of our software, as well as any previous version of our software, are readily available.

We use a decentralized version control system like git. This allows engineers to work on bug fixes, new feature development, and other independent projects simultaneously. Before synchronizing with the central repository, it is recommended that engineers work on local copies created from an appropriate version of the central repository. All changes must be tested before the changes are deployed to users.

Initiating planned changes

While working on new changes (planned or unplanned), it is recommended that you start a new feature branch like in git. All requirements and specifications of a feature may not be known at the beginning of the development of the feature. One can create new branches off the feature branch to develop sub-features as necessary.

Most feature branches exist on the local systems of developers working on the feature. They need not be synced with the central, company-wide, repository right away. However, when a feature is

considered ready to be used by customers, a pull request is created. Any developer or software engineer can initiate a pull request.

Approving planned changes

A pull request outlines the differences in code that this feature proposes. A pull request has to be reviewed by other peer developers or managers. All pull requests should be reviewed and approved by someone who is not the author of the changes. It is recommended that a pull request be reviewed by someone who has expertise in the area where the changes are proposed.

It is recommended that automated triggers, like unit tests, be integrated with pull requests. That is, a pull request might automatically prompt an automated set of tests to run on the changed code, indicating whether it passes some basic safety checks. Other such checks might include code-quality, code linting, or code style checks. Results of such checks are recommended to be logged by the change management system.

Before approving and merging a pull request, the reviewer checks that all prerequisites are met. Typical checks that the reviewer is encouraged to perform are listed below. Not all items in the list are necessary (depending on the type of change), nor is the list exhaustive. Please use your judgment to determine what is necessary, depending on the change under review:

1. Does the proposed change solve the problem it set out to solve? Are all requirements of solving the problem met? If not, were reasonable trade-offs made?
2. Are there any unintended consequences of this change to other parts of the system?
3. Does the change adversely affect any related or unrelated user experience?
4. Are there any algorithmic or logical errors in the proposed change?
5. Does the proposed change require changes in the environment itself (like adding production environment variables etc.)?
6. Could the change create performance issues for itself (or other parts of the system)?
7. Could the proposed change be achieved in a more extensible, robust, or less disruptive way?
8. Does this change introduce any security vulnerabilities?

Unplanned Changes

Sometimes, it becomes necessary to apply unplanned changes, like hotfixes, to the production system to maintain AutoRx Solutions's operational effectiveness. This is usually done to address a situation where the production system is in an undesired state - either from a customer-experience standpoint (like critical bugs, system-down, etc) or from a security standpoint.

Depending on the urgency of the fix required, unplanned changes may skip the requirements of a peer review. Such requests are peer-reviewed post facto. Since we use a version control system, emergency changes can be rolled back if it has unintended or undesirable consequences.

Non-Compliance

Any staff member found to violate this policy may be subject to disciplinary action, which may include termination of employment, or contractual agreement. The action will depend on the extent,

intent, and repercussions of the specific violation(s).

Questions

If you have any questions regarding this policy, please reach out to the policy owner.

End of Change Management Policy.For version history, please see the next page.

Version History

	Version	Log	Date
1	Current	Policy version approved by Abdullah Adeeb	02 Dec, 2025
1		New policy version created	02 Dec, 2025