Here’s a detailed documentation on how Microsoft evaluates new extensions for the Visual Studio Code marketplace, based on the analysis of their practices and tools used.

---

# Evaluation Guidelines and Criteria for Visual Studio Code Marketplace Extensions

## Introduction

This document provides a comprehensive overview of the evaluation guidelines and criteria used by Microsoft for onboarding new Visual Studio Code (VS Code) extensions to their marketplace. The information presented is based on a detailed analysis of the processes and tools employed by Microsoft. This document is intended to guide the implementation of similar standards for a private VS Code marketplace, ensuring high-quality, secure, and compliant extensions.

## Evaluation Criteria

### 1. Security and Vulnerability Scans

Ensuring that extensions do not introduce security risks is paramount. Microsoft conducts extensive security checks using various tools.

#### How it Works:

- \*\*Static Code Analysis:\*\*

- \*\*Tools Used:\*\* [SonarQube](https://docs.sonarqube.org/latest/), [Fortify Static Code Analyzer](https://www.microfocus.com/documentation/fortify-static-code-analyzer-and-tools/)

- \*\*Process:\*\* These tools analyze the source code for vulnerabilities, unsafe coding practices, deprecated APIs, and common security issues like SQL injection and XSS. The analysis is integrated into the CI/CD pipeline, enabling quick detection and resolution of security flaws.

- \*\*Dependency Scanning:\*\*

- \*\*Tools Used:\*\* [OWASP Dependency-Check](https://owasp.org/www-project-dependency-check/), [Snyk](https://snyk.io/docs/)

- \*\*Process:\*\* The tools scan the extension’s dependencies for known vulnerabilities. This ensures no vulnerable dependencies are included in the extension, reducing the risk of indirect security threats.

- \*\*Malware Detection:\*\*

- \*\*Tool Used:\*\* [Microsoft Defender for Endpoint](https://docs.microsoft.com/en-us/microsoft-365/security/defender-endpoint/microsoft-defender-endpoint?view=o365-worldwide)

- \*\*Process:\*\* Microsoft Defender for Endpoint scans the extension package (.vsix file) for malware signatures and suspicious patterns. This automated scan quickly identifies any malicious code within the extension.

### 2. Compliance Checks

Extensions must comply with privacy laws, Microsoft's policies, and industry standards.

#### How it Works:

- \*\*Privacy Policy Compliance:\*\*

- \*\*Reference:\*\* [Microsoft Privacy Statement](https://privacy.microsoft.com/en-us/privacystatement)

- \*\*Process:\*\* Extensions are reviewed to ensure they adhere to relevant privacy laws and Microsoft's privacy policies, including appropriate data handling practices.

- \*\*Terms of Service:\*\*

- \*\*Reference:\*\* [VS Code Marketplace Terms of Use](https://aka.ms/vscode-marketplace-toi)

- \*\*Process:\*\* Extensions are verified for compliance with the marketplace’s terms of service, ensuring they do not engage in prohibited activities.

- \*\*Open Source Licenses:\*\*

- \*\*Tools Used:\*\* [FOSSA](https://fossa.com/), [WhiteSource](https://www.whitesourcesoftware.com/)

- \*\*Process:\*\* These tools review the licenses of open-source components to ensure compatibility with policies and to confirm there are no restrictive conditions on distribution.

### 3. Functional Testing

Functional testing verifies that the extension performs as intended and meets user requirements.

#### How it Works:

- \*\*Basic Feature Testing:\*\*

- \*\*Tools Used:\*\* Automated tests and manual testing procedures.

- \*\*Process:\*\* Automated tests validate that the extension’s primary features work correctly. Manual testing ensures usability and functionality in various scenarios.

- \*\*User Interface Testing:\*\*

- \*\*Tools Used:\*\* [Puppeteer](https://pptr.dev/), [Selenium](https://www.selenium.dev/documentation/en/)

- \*\*Process:\*\* Automated UI tests check for issues related to user interface elements and interactions, ensuring the extension is intuitive and user-friendly.

- \*\*Integration Testing:\*\*

- \*\*Tools Used:\*\* Integration tests using frameworks compatible with VS Code extensions.

- \*\*Process:\*\* The extension is tested for seamless integration with VS Code and other extensions or tools it interacts with, identifying compatibility issues.

### 4. Accessibility Testing

Ensuring accessibility for all users, including those with disabilities, is a priority.

#### How it Works:

- \*\*WCAG Compliance:\*\*

- \*\*Tools Used:\*\* [Axe Accessibility](https://www.deque.com/axe/), [WAVE](https://wave.webaim.org/)

- \*\*Process:\*\* Extensions are reviewed against Web Content Accessibility Guidelines (WCAG) to ensure they are accessible to users with disabilities.

- \*\*Keyboard Navigation:\*\*

- \*\*Process:\*\* Extensions are tested to ensure full functionality via keyboard, allowing users who cannot use a mouse to navigate and use the extension.

- \*\*Screen Reader Compatibility:\*\*

- \*\*Tools Used:\*\* [JAWS](https://www.freedomscientific.com/products/software/jaws/), [NVDA](https://www.nvaccess.org/)

- \*\*Process:\*\* Testing with screen readers ensures a good experience for visually impaired users, verifying that all UI elements are properly labeled and announced.

### 5. Post-Publication Monitoring

Microsoft continues to monitor extensions even after publication to ensure ongoing compliance and security.

#### How it Works:

- \*\*User Feedback:\*\*

- \*\*Process:\*\* Continuously collecting and reviewing user feedback to identify and address any issues.

- \*\*Automated Monitoring:\*\*

- \*\*Tools Used:\*\* Various monitoring tools integrated with the marketplace.

- \*\*Process:\*\* Automated tools detect anomalies or new security vulnerabilities post-publication, ensuring ongoing security and performance.

- \*\*Regular Audits:\*\*

- \*\*Process:\*\* Periodic audits of extensions to ensure they continue to comply with policies and standards.

## Submission Process

1. \*\*Initial Submission:\*\*

- Developers submit their extension for review through the VS Code marketplace portal, including the extension package, documentation, and any required compliance information.

2. \*\*Automated Scans:\*\*

- Initial automated scans are performed to detect any immediate issues or vulnerabilities.

3. \*\*Manual Review:\*\*

- A team of reviewers conducts a thorough manual review of the extension, focusing on security, functionality, and compliance.

4. \*\*Feedback and Iteration:\*\*

- Developers may receive feedback and requests for modifications if any issues are detected. They address the feedback and resubmit the extension for further review.

5. \*\*Approval and Publication:\*\*

- Once the extension passes all checks and reviews, it is approved and published on the VS Code marketplace.

6. \*\*Post-Publication Monitoring:\*\*

- Continuous monitoring and regular audits to ensure ongoing compliance and security.

## Conclusion

Implementing these guidelines and evaluation criteria ensures that extensions in our private VS Code marketplace meet high standards for security, performance, compliance, and user experience. This rigorous process helps maintain the quality and trustworthiness of the extensions available to our users.

## References

- [VS Code Marketplace Policies](https://aka.ms/vscode-marketplace-policies)

- [Microsoft Security Best Practices](https://learn.microsoft.com/en-us/security/compass/)

- [Web Content Accessibility Guidelines (WCAG)](https://www.w3.org/WAI/standards-guidelines/wcag/)

---

You can copy and paste this content into your Confluence page and format it as needed to match your organization’s style and requirements. This version provides detailed and specific information about the tools and processes used by Microsoft for evaluating VS Code extensions, tailored to your private marketplace.

# Evaluation Guidelines and Criteria for Visual Studio Code Marketplace Extensions

## Introduction

This document provides a comprehensive overview of the evaluation guidelines and criteria used by Microsoft for onboarding new Visual Studio Code (VS Code) extensions to their marketplace. The information presented is based on a detailed analysis of the processes and tools employed by Microsoft. This document is intended to guide the implementation of similar standards for a private VS Code marketplace, ensuring high-quality, secure, and compliant extensions.

## Evaluation Criteria

### 1. Security and Vulnerability Scans

Ensuring that extensions do not introduce security risks is paramount. Microsoft conducts extensive security checks using various tools.

#### How it Works:

**Static Code Analysis:**

* + **Tools Used:** SonarQube, Fortify Static Code Analyzer
  + **Process:** These tools analyze the source code for vulnerabilities, unsafe coding practices, deprecated APIs, and common security issues like SQL injection and XSS. The analysis is integrated into the CI/CD pipeline, enabling quick detection and resolution of security flaws.

**Dependency Scanning:**

* + **Tools Used:** OWASP Dependency-Check, Snyk
  + **Process:** The tools scan the extension’s dependencies for known vulnerabilities. This ensures no vulnerable dependencies are included in the extension, reducing the risk of indirect security threats.

**Malware Detection:**

* + **Tool Used:** [Microsoft Defender for Endpoint](https://docs.microsoft.com/en-us/microsoft-365/security/defender-endpoint/microsoft-defender-endpoint?view=o365-worldwide" \t "_new)
  + **Process:** Microsoft Defender for Endpoint scans the extension package (.vsix file) for malware signatures and suspicious patterns. This automated scan quickly identifies any malicious code within the extension.

### 2. Compliance Checks

Extensions must comply with privacy laws, Microsoft's policies, and industry standards.

#### How it Works:

**Privacy Policy Compliance:**

* + **Reference:** [Microsoft Privacy Statement](https://privacy.microsoft.com/en-us/privacystatement" \t "_new)
  + **Process:** Extensions are reviewed to ensure they adhere to relevant privacy laws and Microsoft's privacy policies, including appropriate data handling practices.

**Terms of Service:**

* + **Reference:** VS Code Marketplace Terms of Use
  + **Process:** Extensions are verified for compliance with the marketplace’s terms of service, ensuring they do not engage in prohibited activities.

**Open Source Licenses:**

* + **Tools Used:** [FOSSA](https://fossa.com/" \t "_new), [WhiteSource](https://www.whitesourcesoftware.com/" \t "_new)
  + **Process:** These tools review the licenses of open-source components to ensure compatibility with policies and to confirm there are no restrictive conditions on distribution.

### 3. Functional Testing

Functional testing verifies that the extension performs as intended and meets user requirements.

#### How it Works:

**Basic Feature Testing:**

* + **Tools Used:** Automated tests and manual testing procedures.
  + **Process:** Automated tests validate that the extension’s primary features work correctly. Manual testing ensures usability and functionality in various scenarios.

**User Interface Testing:**

* + **Tools Used:** [Puppeteer](https://pptr.dev/" \t "_new), Selenium
  + **Process:** Automated UI tests check for issues related to user interface elements and interactions, ensuring the extension is intuitive and user-friendly.

**Integration Testing:**

* + **Tools Used:** Integration tests using frameworks compatible with VS Code extensions.
  + **Process:** The extension is tested for seamless integration with VS Code and other extensions or tools it interacts with, identifying compatibility issues.

### 4. Accessibility Testing

Ensuring accessibility for all users, including those with disabilities, is a priority.

#### How it Works:

**WCAG Compliance:**

* + **Tools Used:** Axe Accessibility, WAVE
  + **Process:** Extensions are reviewed against Web Content Accessibility Guidelines (WCAG) to ensure they are accessible to users with disabilities.

**Keyboard Navigation:**

* + **Process:** Extensions are tested to ensure full functionality via keyboard, allowing users who cannot use a mouse to navigate and use the extension.

**Screen Reader Compatibility:**

* + **Tools Used:** JAWS, [NVDA](https://www.nvaccess.org/" \t "_new)
  + **Process:** Testing with screen readers ensures a good experience for visually impaired users, verifying that all UI elements are properly labeled and announced.

### 5. Post-Publication Monitoring

Microsoft continues to monitor extensions even after publication to ensure ongoing compliance and security.

#### How it Works:

**User Feedback:**

* + **Process:** Continuously collecting and reviewing user feedback to identify and address any issues.

**Automated Monitoring:**

* + **Tools Used:** Various monitoring tools integrated with the marketplace.
  + **Process:** Automated tools detect anomalies or new security vulnerabilities post-publication, ensuring ongoing security and performance.

**Regular Audits:**

* + **Process:** Periodic audits of extensions to ensure they continue to comply with policies and standards.

## Submission Process

**Initial Submission:**

* + Developers submit their extension for review through the VS Code marketplace portal, including the extension package, documentation, and any required compliance information.

**Automated Scans:**

* + Initial automated scans are performed to detect any immediate issues or vulnerabilities.

**Manual Review:**

* + A team of reviewers conducts a thorough manual review of the extension, focusing on security, functionality, and compliance.

**Feedback and Iteration:**

* + Developers may receive feedback and requests for modifications if any issues are detected. They address the feedback and resubmit the extension for further review.

**Approval and Publication:**

* + Once the extension passes all checks and reviews, it is approved and published on the VS Code marketplace.

**Post-Publication Monitoring:**

* + Continuous monitoring and regular audits to ensure ongoing compliance and security.

## Conclusion

Implementing these guidelines and evaluation criteria ensures that extensions in our private VS Code marketplace meet high standards for security, performance, compliance, and user experience. This rigorous process helps maintain the quality and trustworthiness of the extensions available to our users.

## References

* VS Code Marketplace Policies
* [Microsoft Security Best Practices](https://learn.microsoft.com/en-us/security/compass/" \t "_new)
* [Web Content Accessibility Guidelines (WCAG)](https://www.w3.org/WAI/standards-guidelines/wcag/" \t "_new)

## **Microsoft Visual Studio Code Extension Evaluation Criteria**

When publishing a new extension to the Visual Studio Code marketplace, Microsoft evaluates the extension through a series of automated and manual checks to ensure its quality, security, and compatibility. The following documentation outlines the evaluation criteria and processes employed by Microsoft.

## **Automated Checks**

## 1. **Vulnerability Checks**

Microsoft uses the [OWASP Zed Attack Proxy (ZAP)]((link unavailable)) to scan extensions for potential security vulnerabilities, including:

SQL injection

Cross-site scripting (XSS)

Cross-site request forgery (CSRF)

## 2. **Malware Detection**

Microsoft utilizes the [Windows Defender Advanced Threat Protection (ATP)]((link unavailable)) to scan extensions for malware and other malicious software.

## 3. **Compatibility Checks**

Extensions are verified to be compatible with the latest versions of Visual Studio Code, including:

Version compatibility

Platform compatibility (Windows, macOS, Linux)

## 4. **Package Validation**

The extension's package is validated for correctness and consistency, including:

Package format (.vsix)

Package metadata (manifest, icons, etc.)

## **Manual Checks**

## 1. **Code Review**

Microsoft performs a manual review of the extension's code to ensure it meets the Visual Studio Code extension guidelines and best practices, including:

Code organization and structure

Performance and efficiency

Security and privacy

## 2. **Functional Testing**

Extensions are manually tested to ensure they function as described and meet the expected user experience, including:

Feature testing

User interface testing

## 3. **Content Review**

Microsoft reviews the extension's content, including:

Extension description and metadata

Icon and branding consistency

Compliance with Visual Studio Code marketplace policies

## **Additional Information**

Microsoft reserves the right to manually review and test extensions even after successful publication to ensure ongoing quality and security.

Extensions may be removed from the marketplace if they are found to violate the evaluation criteria or guidelines.

## **References**

[Visual Studio Code Extension Guidelines]((link unavailable))

[Visual Studio Code Marketplace Policies]((link unavailable))

By understanding these evaluation criteria and processes, you can ensure your extensions meet the necessary standards for publication in the Visual Studio Code marketplace.