



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

## Praxistransferbericht

---

# Einbindung von Code Compliance Checks in der SAP Software Entwicklung

---

vorgelegt am 24. März 2024

<b>Name:</b>	Justin Becker
<b>Matrikelnummer:</b>	77204368351
<b>Ausbildungsbetrieb:</b>	SAP SE
<b>Fachbereich:</b>	FB2: Duales Studium — Technik
<b>Studienjahrgang:</b>	2023
<b>Studiengang:</b>	Informatik
<b>Betreuerin Unternehmen:</b>	Jenny Peter
<b>Betreuerin Hochschule:</b>	Lara Maria Stricker
<b>Wortanzahl:</b>	2202

Von der betrieblichen Betreuung zur Kenntnis genommen:

---

Jenny Peter

---

Justin Becker

# Zusammenfassung

Das Ziel dieser Arbeit besteht darin, den Entwicklungsprozess von SAP-Software zu vereinfachen, indem die Dienste des HANA & HANA Cloud Quality Engineering Teams der Common Service Infrastructure (CSI) Abteilung verbessert werden. Diese Dienste stellen sogenannte Code Compliance, also ob Code den SAP-Richtlinien folgt, fest. Diese Dienste sichern die Code Compliance, d. h. die technischen Richtlinien von SAP ab.

Dieser Bericht baut darauf, wie die Bewertung der Scan Ergebnisse auf einer technischen Ebenen so gestaltet, dass sie so streng wie nötig und so tolerant wie möglich sind. Dabei soll die Verantwortung für die Bewertung klar in dafür zuständigen Händen liegen.

# Inhaltsverzeichnis

<b>Zusammenfassung</b>	<b>I</b>
<b>Inhaltsverzeichnis</b>	<b>II</b>
<b>Abbildungsverzeichnis</b>	<b>III</b>
<b>1 Einleitung</b>	<b>1</b>
<b>2 Softwareentwicklung bei SAP</b>	<b>2</b>
2.1 Cloud Produkte . . . . .	3
2.2 CI/CD . . . . .	3
<b>3 Code Compliance</b>	<b>5</b>
3.1 Beispiel . . . . .	5
3.2 Checks . . . . .	6
3.2.1 Scans . . . . .	6
3.2.2 Scan-Ergebnis-Bewertung . . . . .	7
<b>4 Herausforderungen</b>	<b>9</b>
4.1 Externe Tools . . . . .	9
4.2 Code Ausführung in Pipelines . . . . .	10
<b>5 Fazit</b>	<b>11</b>
<b>Ehrenwörtliche Erklärung</b>	<b>12</b>

# Abbildungsverzeichnis

1	Produktportfolio Übersicht von SAP . . . . .	2
2	Python Risikocode, da eine geöffnete Datei nicht richtig geschlossen wird	5
3	Beispiel Rego Script mit Ein- und Ausgabe . . . . .	7
4	Beispiel Script für Code, der nicht in einer Pipeline funktioniert . . . .	10
5	Beispiel Script für Code, der Pipeline Funktionalität benutzt . . . . .	10

## Hinweis

Aus Gründen der besseren Lesbarkeit wird im Text verallgemeinernd die männliche Form verwendet. Diese Formulierungen umfassen gleichermaßen weibliche und männliche Personen.

# Glossar

**JSON** von Programmiersprachen unabhängiges Format zum Speichern von Daten durch eine Key-Value-Pair-Zuweisung

**Pipeline** Abfolge von Prozessen, die aufeinander aufbauen

**Policy As Code** Prinzip, Richtlinien als Code mit einer Programmiersprache aufzuschreiben

**Public Cloud** Cloud, die von mehreren Unternehmen genutzt wird

**Software as a Service** komplettes Softwareprogramm in der Cloud mit Nutzung über einen Browser

# Akronyme

**CCC** Code Compliance Check

**CI/CD** Continuous Integration / Continuous Delivery

**CSI** Common Service Infrastructure

**EU** Europäische Union

**HHCQE** HANA & HANA Cloud Quality Engineering

**OPA** Open Policy Agent

**SaaS** Software as a Service

# 1 Einleitung

Im Unternehmen SAP SE besteht die Notwendigkeit, dass alle Software-Entwickler die globale Entwicklungsrichtlinie von SAP und insbesondere die in dieser Richtlinie festgelegten Regeln befolgen. Die Einhaltung dieser Regeln und damit die Einhaltung der SAP Global Development-Richtlinie für alle Entwicklungseinheiten ist ungemein wichtig und obligatorisch, um jegliche rechtlichen und finanziellen Risiken, sowie Ansehensverluste für das Unternehmen zu vermeiden.

Die SAP-Abteilung CSI unterstützt Service- und Produktteams beim Programmieren, Versenden und Ausführen ihrer Services und Produkte in der Cloud. Sie bietet ein Portfolio von Diensten, die eine Plattform für die Entwicklung, Veröffentlichung und den Betrieb von Cloud-nativen, konformen und produktionsbereiten Diensten und Anwendungen bilden.

Die Dienste zur Überprüfung der SAP-Entwicklungsrichtlinien werden vom Team HANA & HANA Cloud Quality Engineering (HHCQE) entwickelt und betreut. Mit sogenannten Code Compliance Checks (CCCs) werden die Richtlinien in automatisierten Entwicklungs- und Bereitstellungsprozessen überprüft und die SAP-konforme Auslieferung neuer oder aktualisierter Software, z. B. zur Fehlerbehebung, gewährleistet.

In diesem Praxistransferbericht wird erarbeitet, wie Checks gestaltet werden, damit sie alle genannten Anforderungen erfüllen und gleichzeitig so minimalistisch bleiben, dass kein Entwickler mehr als nötig durch negative Check Ergebnisse aufgehalten wird.

## 2 Softwareentwicklung bei SAP

Um den Umfang der Anforderung an die CCCs zu verstehen, muss erst ein Verständnis für die Entwicklung der SAP-Produkte und damit auch für die Produkte selbst entstehen.

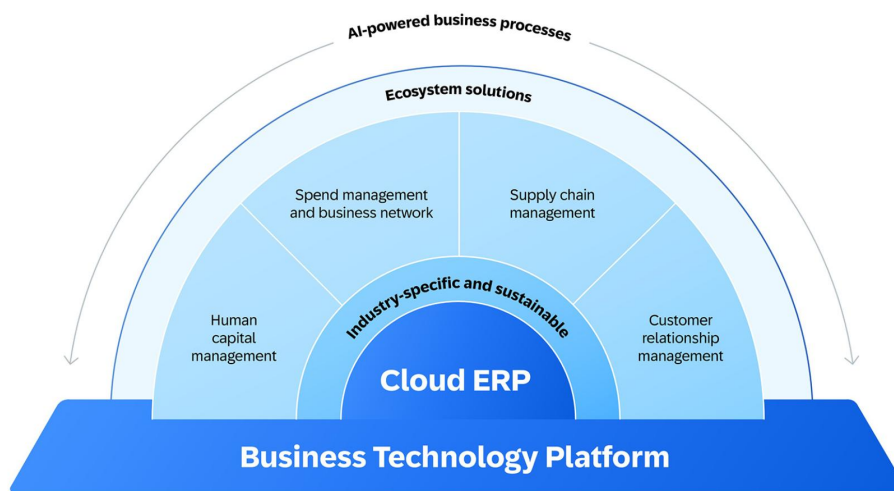


Abbildung 1: Produktportfolio Übersicht von SAP

Quelle: [https://www.sap.com/intelligent-enterprise.html?url\\_id=banner-glo-homepage-row3-cta-who-we-are-230510](https://www.sap.com/intelligent-enterprise.html?url_id=banner-glo-homepage-row3-cta-who-we-are-230510)

Die SAP ist eine der größten Software-Entwicklungsunternehmen weltweit und hat, wie in Abbildung 1 dargestellt, ein sehr breites Produktportfolio für ein sehr breites Kundenfeld [**CorporateFactSheet**]. Diese Produkte umfassen Bereiche wie Human Resources, Customer Relationship Management, branchenspezifische Lösungen und noch vieles mehr.

Damit Kunden in der Benutzung eine klare und einheitliche Handhabung erfahren können, muss jedes Produkt als Softwarekomponente genauso gestaltet sein. Nicht nur das – sämtliche SAP-Produkte sollten kongruent zueinander sein, da bei Kunden nicht das Gefühl entstehen darf, sie müssten mit jedem Produkt etwas komplett Neues erlernen. Dies würde Kunden davon abhalten, in weitere SAP-Produkte zu investieren. [**SAPUsability**]



## 2.1 Cloud Produkte

Die Strategie der SAP sieht vor, dass möglichst viele dieser Produkte in einer Public Cloud als Software as a Service (SaaS) verfügbar sind [**SAPCloudStrategy**]. Die Entwicklung von Cloud Produkten hat unter anderem den Vorteil, dass Kunden nicht in eigene Hardware und Infrastruktur investieren müssen und damit flexibel bleiben. Da Cloud-Produkte meist in einem Abonnement-Modell verkauft werden, sichert sich die SAP konstante und kalkulierbare Einnahmen. Deshalb bauen aktuelle SAP-Produkte, wie in Abbildung 1 zu sehen, auf diesem Prinzip auf. [**CloudProContra**]

Damit die Entwicklung der Produkte den erklärten Prinzipien entsprechend abläuft, gibt es Abteilungen, die sich darauf fokussieren, den Cloud-Entwicklungsprozess zu vereinfachen. So bietet CSI eine einheitliche Infrastruktur für diesen Entwicklungsprozess. Als Teil dieser Infrastruktur kann überprüft werden, ob alle Anforderungen an den Code des Produktes erfüllt sind.

Umgesetzt werden diese Anforderungen als Teil des Continuous Integration / Continuous Delivery (CI/CD) Prozesses.

## 2.2 CI/CD

Da die Cloud-Produkte komplett von SAP verwaltet werden, ergibt sich der wichtige Vorteil für das Unternehmen und deren Kunden, dass die Entwicklungsteams bei SAP den Code von Produkten jederzeit anpassen und an den Kunden ausliefern kann. Somit können schnell Fehler korrigiert, sowie Feedback des Kunden umgesetzt und agil gearbeitet werden. So entsteht mehr Produktwert für die Kunden und ihre Zufriedenheit steigt.

Dieser Prozess wird CI/CD genannt. Jeder Entwickler integriert kontinuierlich neue Änderungen in den Code, welche dann fortlaufend an den Kunden geliefert werden. Umgesetzt wird dieses Prinzip durch die Automatisierung mit sogenannten CI/CD Pipelines. Diese Pipelines übernehmen alle für die Auslieferung nötigen Schritte, wie zum Beispiel die Build-, Test- und Deploy-Vorgänge und ermöglichen damit erst CI/CD. [**CI/CD**]

Durch den zügigen Entwicklungsprozess ist die Fehleranfälligkeit, d. h. dass ein Code nicht der Compliance entspricht, relativ hoch. Deshalb ist es ein unerlässlicher Schritt,

die Code Compliance vor dem Ausliefern als Teil der CI/CD Pipelines festzustellen. Deshalb werden CCCs als solch ein Schritt eingeführt.

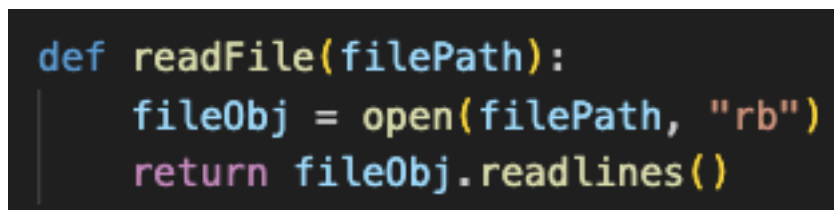
## 3 Code Compliance

Code gilt als compliant, wenn er alle ihn betreffenden Richtlinien erfüllt. Das können Richtlinien von Staatlichen Institutionen wie der EU oder Deutschland sein, Richtlinien, die für das ganze Unternehmen SAP gelten, aber auch Richtlinien, die nur für bestimmte Abteilungen gelten.

Deshalb ist für die CCCs wichtig zu wissen, wie die Richtlinien von SAP genutzt werden können. Diese umfassen unter anderem Regeln zum Qualitätsmanagement, zur Export-Kontrollklassifizierung, Opensource Lizenzvalidierung und zu diversen Überprüfungen der Sicherheitsanforderungen.

### 3.1 Beispiel

Ein konkreter Beispiel-Code, der ein Sicherheitsrisiko darstellt – und damit eventuell für nicht Compliance verantwortlich ist, ist die in Abbildung 2 dargestellte Funktion.



```
def readFile(filePath):  
    fileObj = open(filePath, "rb")  
    return fileObj.readlines()
```

Abbildung 2: Python Risikocode, da eine geöffnete Datei nicht richtig geschlossen wird (selbst geschrieben)

In dem Beispiel wird mit Python eine Datei geöffnet und ihr Inhalt gelesen. Die Datei wird allerdings nicht geschlossen. Solche Fehler könnten zu unerwartetem Verhalten während der Laufzeit führen. Deshalb müssen sie bereits davor erkannt werden.

Da eine Überprüfung des gesamten Codes per Hand fehleranfällig und langsam ist, braucht es hierfür eine Automatisierung. Diese Automatisierung wird Check genannt.

### 3.2 Checks

Damit ergibt sich als Ziel der CCCs in den CI/CD Pipelines, dass sie das Ergebnis liefern sollen, ob der Code, der mit der Pipeline ausgeliefert werden soll, allen erforderlichen Richtlinien folgt. Außerdem müssen sie eine Begründung für diese Entscheidung an die Entwickler liefern, damit dieser seinen Code entsprechend verbessern kann.

Dafür sind CCCs in zwei Teile aufgeteilt:

- ein Scan, der den Code analysiert und feststellt, welche Richtlinien relevante Risikostellen existieren
- die Bewertung der Scan-Ergebnisse nach den Richtlinien

Da die Richtlinien Anforderungen unterschiedlichster Themenbereiche von der Export-Kontrollklassifizierung bis zur Sicherheit enthalten, wird die Gesamtfeststellung - compliant oder nicht - in mehrere Teilchecks unterteilt, sodass ein Check immer nur einen Aspekt der Richtlinien übernimmt.

#### 3.2.1 Scans

Das Scannen von Code wird von externen Tools übernommen. Zum Aufrufen dieser Tools wird eine interne Version des Project Piper genutzt. Piper vereinfacht unterschiedlichste Schritte, die in einer Pipeline ausgeführt werden können. So beispielsweise auch das Aufrufen von externen Scan Tools.

Es gibt im Bereich der Sicherheitsanalyse von Code eine Reihe geeigneter Tools wie Checkmarx oder Fortify, die unter anderem vom HHCQE-Team genutzt werden [SASTTools]. Solche Tools würden eine Risikostelle wie in Abbildung 2 erkennen und als „Improper\_Resource\_Shutdown\_or\_Release“ einordnen [SecurityVulnerabilities].

Von dem Scan wird eine Datei erzeugt, die die gefundenen Risikostellen enthält. Zudem enthält die Datei meist eine Priorisierung, wie schwerwiegend die Risikostellen sind.

Diese Datei stellt nun das Ergebnis des Scans dar und liegt im JSON-Format vor.

Außerdem besitzen externe Tools wie Checkmarx oft eine Weboberfläche, in der Details eines Scans mit den genauen Zeilen an Code, die Risikostellen sind, eingesehen werden können. In diesem Rahmen gibt es die Möglichkeit, diese Risikostellen als unbedenklich zu markieren. Richtlinien beziehen sich oft darauf, wie viele von den Risikostellen als unbedenklich eingestuft werden müssen.

#### 3.2.2 Scan-Ergebnis-Bewertung

In der Bewertung der Ergebnisse muss nun auf Grundlage der erzeugten Datei die Entscheidung getroffen werden, ob der Code compliant ist oder nicht. Bisher wurde diese Entscheidung nicht eindeutig nach den Richtlinien getroffen. Stattdessen wurde sie weitgehend von Piper übernommen. Das Problem dabei ist, dass dieser Prozess mit Piper nicht transparent ist und keine Verantwortungen abgeklärt sind. Es bedarf einer flexiblen, nachvollziehbaren Lösung, falls sich die Anforderungen an Code und damit die Richtlinien ändern, sodass immer klar ist, auf welcher Grundlage die Entscheidung getroffen wurde.

Von der SAP Abteilung Hyperspace werden seit Neustem Richtlinien in einem Policy As Code Format bereitgestellt. Diese Richtlinien sind in der sogenannten Rego-Programmiersprache formuliert, die mit dem Open Policy Agent (OPA) angewandt werden können. Dieser OPA wird von Hyperspace genutzt.

<pre>package example import rego.v1  default compliant := false compliant if {     input.vulnerabilities &lt; 10 }</pre>	<table><tr><th colspan="2">INPUT</th></tr><tr><td>1</td><td>{</td></tr><tr><td>2</td><td>    "vulnerabilities": 10</td></tr><tr><td>3</td><td>}</td></tr><tr><th colspan="2">DATA</th></tr><tr><th colspan="2">OUTPUT</th></tr><tr><td colspan="2">Found 1 result in 89µs.</td></tr><tr><td>1</td><td>{</td></tr><tr><td>2</td><td>    "compliant": false</td></tr><tr><td>3</td><td>}</td></tr></table>	INPUT		1	{	2	"vulnerabilities": 10	3	}	DATA		OUTPUT		Found 1 result in 89µs.		1	{	2	"compliant": false	3	}
INPUT																					
1	{																				
2	"vulnerabilities": 10																				
3	}																				
DATA																					
OUTPUT																					
Found 1 result in 89µs.																					
1	{																				
2	"compliant": false																				
3	}																				

Abbildung 3: Beispiel Rego Script mit Ein- und Ausgabe  
(selbst geschrieben)

Das in Abbildung 3 zu sehende Script ist immer eindeutig, transparent und schnell anzupassen. Die verwendete Rego-Programmiersprache ist außerdem dafür optimiert, Richtlinien anzuwenden. Damit ist der OPA mit Rego für die genannten Anforderungen geeignet. [Rego]

Genutzt werden können diese Richtlinien von Hyperspace in zwei Schritten.

Als erstes wird die Ergebnisdatei des Scans auf einen Server von Hyperspace hochgeladen. In dem Upload ist noch ein sogenannter Policy Key (Deutsch: Richtlinien-Schlüssel) enthalten, der eindeutig zuordnet, welche Richtlinie für das Ergebnis genutzt werden soll. So können die Ergebnisse des Scans bei Hyperspace mit Hilfe von OPA ausgewertet werden.

Im zweiten Schritt müssen die Auswertungsergebnisse innerhalb der Pipeline nutzbar gemacht werden. Dafür wird eine Anfrage an den Hyperspace-Server geschickt, der eine Datei mit der Auswertung zurückgibt. Diese Datei enthält im JSON-Format die Information, ob der gescannte Code der Richtlinie mit gegebenen Policy Key entspricht oder nicht.

Da die Evaluation auf dem Hyperspace Server asynchron läuft und nicht klar ist, wann die Auswertung zur Verfügung steht, wird ein Fallback-Mechanismus implementiert, der diese zweite Anfrage in größer werdenden Zeitabständen bis zu fünf Mal wiederholt. Wenn es nach dem fünften Versuch kein Ergebnis gibt, bricht die Bewertung ab und das Check-Gesamtergebnis wird als unbekannt gesetzt.

Damit für den Endbenutzer dieser Entscheidungsfindungsprozess nachvollziehbar bleibt, werden mit einer dritten Anfrage an den Hyperspace Server Details zu der Auswertung abgerufen. Diese Details werden dem Benutzer zusätzlich zu dem eigentlichen Auswertungsergebnis - compliant oder nicht - angezeigt. Sie enthalten Informationen wie den Namen der Richtlinie, eine Kurzbeschreibung, die URL zu der genutzten Richtlinie im Policy As Code Format sowie noch einiges mehr.

## 4 Herausforderungen

Während der Erarbeitung dieses Projekts sind immer wieder Probleme aufgetreten, die diese Arbeit wohl erst interessant machen. Neben alltäglichen Problemen mit dem Code gab es zwei Herausforderungen, die besonders lehrreich waren.

### 4.1 Externe Tools

Im Verlauf der Praxisphase wurde mit einer breiten Auswahl an Tools von anderen Teams gearbeitet. Dazu gehörten zum Beispiel das Project Piper, interne GitHub-Varianten und unter anderem auch die Richtlinien, die von dem Hyperspace-Team zur Verfügung gestellt wurden.

Zu Beginn der Praxisphase wurde ein Plan zur Nutzung der Hyperspace-Richtlinien aufgestellt, der vorgab, dass die Evaluation der Scan-Ergebnisse direkt in der Pipeline stattfindet und nicht bei Hyperspace. Dafür sollten die Richtlinien im Policy As Code Format von einem Hyperspace GitHub Repository in den Pipeline Workspace heruntergeladen werden. Danach wurde die Ergebnisdatei des Scans mit OPA gegen die heruntergeladene Richtlinie lokal evaluiert. Dieser Vorgang hat den Vorteil, dass die Auswertungsergebnisse zuverlässig und schnell verfügbar sind.

In der Vorstellung dieses Prozesses nach guten zwei Wochen Entwicklung der Anwendung für die neu verfügbaren Policy As Code Richtlinien gegenüber Hyperspace, wurde vom Hyperspace Team angemerkt, dass diese Idee nicht skalieren wird. Der Grund dafür war, dass nicht alle Richtlinien wie angenommen in einem GitHub Repository zur Verfügung stehen werden, sondern verteilt sind. So gibt es keinen zentralen Anlaufpunkt von dem man Richtlinien herunterladen könnte.

Deshalb wurde der in diesem Bericht unter Kapitel 3.2.2 vorgestellte Lösungsansatz umgesetzt. Mitzunehmen aus dieser Erfahrung ist, dass man bei der Nutzung von externen Tools möglichst viel über dessen Funktionsweise herausfinden sollte. Direkte Kommunikation und Absprache mit den Teams über die Nutzung kann viel Zeit sparen und ist eine Möglichkeit, dieses Problem zu vermeiden.

### 4.2 Code Ausführung in Pipelines

Bei der Entwicklung von dem Feature wurde viel lokal getestet, da Tests in Pipelines viel Zeit kosten. So war eine Herausforderung, dass sich die geschriebenen Code-Skripte in den beiden Umgebungen unterschiedlich verhalten.

```
String resultAsText = new File(POLICY_COLLECT_RESULT_FILE_PATH).getText()
```

Abbildung 4: Beispiel Script für Code, der nicht in einer Pipeline funktioniert  
(selbst geschrieben)

In dem Beispiel-Script in Abbildung 4 ist zu sehen, wie mit einer Standard-Java-Funktionalität die Auswertungsergebnisdatei gelesen wird. Diese Implementierung funktioniert lokal problemlos, scheitert aber in einer Pipeline, da diese Java Funktionalität nicht in der Pipeline Umgebung verfügbar ist.

```
String resultAsText = PipelineEnvironment.instance.getScript().readFile([file: POLICY_COLLECT_RESULT_FILE_PATH])
```

Abbildung 5: Beispiel Script für Code, der Pipeline Funktionalität benutzt  
(selbst geschrieben)

Um dieses Problem zu umgehen, stehen Funktionalitäten von der Pipeline-Umgebung zur Verfügung. Der Aufruf aus Abbildung 5 zeigt wie, so eine Funktionalität benutzt werden kann.

Bei dieser Implementierung ist zu beachten, dass sie vorerst nur in der Pipeline funktioniert. Lokal besteht die Möglichkeit bei diesem Aufruf im Hintergrund den Code aus Abbildung 4 auszuführen, damit auch weiterhin getestet werden kann. Alle Code Abschnitte, die mit Dateien arbeiten, mussten entsprechend angepasst werden.

Durch diese Herausforderung ist das Verständnis für die Funktionsweise von Pipelines und das Bewusstsein für die Bedeutung der Umgebung, in der gearbeitet wird, gewachsen.



## 5 Fazit

Abschließend lässt sich festhalten, dass die praktisch umgesetzte Implementierung alle theoretisch erarbeiteten Anforderungen erfüllt. D. h., dass innerhalb einer Pipeline durch einen Upload zu einem Hyperspace-Server und einem anschließenden Download der Auswertungsergebnisse entschieden werden kann, ob der Code compliant ist oder nicht. Der Entscheidungsfindungsprozess wird durch von dafür Verantwortlichen in dem unmissverständlichen Policy As Code Format deutlich gemacht. Für tiefere Einblicke kann in den genutzten Scan-Tools eine Weboberfläche genutzt werden, die alle Risikostellen dokumentiert und einordnet.

So kann der rasante CI/CD-Entwicklungsprozess für die SAP-Cloud abgesichert werden, damit SAP Kunden durchgehend hochqualitative Produkte erhalten.

Trotzdem bleiben noch Fragen offen. Was machen die Teams, die eigene strengere oder mildere Anforderungen haben? Gibt es gegebenenfalls Richtlinien, die gar nicht zentral in einem Policy As Code Format dargelegt werden können? Wie sieht die Bewertung mit Richtlinien aus, wenn von einem Scan keine eindeutige JSON-Ergebnisdatei erzeugt wird?

Wie zu erkennen, lässt dieses Feld nach ersten Anstrengungen weiterhin viel Raum zur Entwicklung und Einbindung von verbesserten Code Compliance Checks in der SAP-Softwareentwicklung.

# Ehrenwörtliche Erklärung

Ich erkläre ehrenwörtlich:

1. dass ich meinen Praxistransferbericht selbstständig verfasst habe,
2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe,
3. dass ich meinen Praxistransferbericht bei keiner anderen Prüfung vorgelegt habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

---

Ort, Datum

---

Justin Becker