Projekt Check für TIA Portal

TIA Portal / Openness API / Programmierstyleguide

https://support.industry.siemens.com/cs/ww/de/view/109741418

Siemens Industry Online Support

Rechtliche Hinweise

Nutzung der Anwendungsbeispiele

In den Anwendungsbeispielen wird die Lösung von Automatisierungsaufgaben im Zusammenspiel mehrerer Komponenten in Form von Text, Grafiken und/oder Software-Bausteinen beispielhaft dargestellt. Die Anwendungsbeispiele sind ein kostenloser Service der Siemens AG und/oder einer Tochtergesellschaft der Siemens AG ("Siemens"). Sie sind unverbindlich und erheben keinen Anspruch auf Vollständigkeit und Funktionsfähigkeit hinsichtlich Konfiguration und Ausstattung. Die Anwendungsbeispiele stellen keine kundenspezifischen Lösungen dar, sondern bieten lediglich Hilfestellung bei typischen Aufgabenstellungen. Sie sind selbst für den sachgemäßen und sicheren Betrieb der Produkte innerhalb der geltenden Vorschriften verantwortlich und müssen dazu die Funktion des jeweiligen Anwendungsbeispiels überprüfen und auf Ihre Anlage individuell anpassen.
Sie erhalten von Siemens das nicht ausschließliche, nicht unterlizenzierbare und nicht übertragbare Recht, die Anwendungsbeispiele durch fachlich geschultes Personal zu nutzen. Jede Änderung an den Anwendungsbeispielen erfolgt auf Ihre Verantwortung. Die Weitergabe an Dritte oder Vervielfältigung der Anwendungsbeispiele oder von Auszügen daraus ist nur in Kombination mit Ihren eigenen Produkten gestattet. Die Anwendungsbeispiele unterliegen nicht zwingend den üblichen Tests und Qualitätsprüfungen eines kostenpflichtigen Produkts, können Funktions- und Leistungsmängel enthalten und mit Fehlern behaftet sein. Sie sind verpflichtet, die Nutzung so zu gestalten, dass eventuelle Fehlfunktionen nicht zu Sachschäden oder der Verletzung von Personen führen.

Haftungsausschluss

Siemens schließt seine Haftung, gleich aus welchem Rechtsgrund, insbesondere für die Verwendbarkeit, Verfügbarkeit, Vollständigkeit und Mangelfreiheit der Anwendungsbeispiele, sowie dazugehöriger Hinweise, Projektierungs- und Leistungsdaten und dadurch verursachte Schäden aus. Dies gilt nicht, soweit Siemens zwingend haftet, z.B. nach dem Produkthaftungsgesetz, in Fällen des Vorsatzes, der groben Fahrlässigkeit, wegen der schuldhaften Verletzung des Lebens, des Körpers oder der Gesundheit, bei Nichteinhaltung einer übernommenen Garantie, wegen des arglistigen Verschweigens eines Mangels oder wegen der schuldhaften Verletzung wesentlicher Vertragspflichten. Der Schadensersatzanspruch für die Verletzung wesentlicher Vertragspflichten ist jedoch auf den vertragstypischen, vorhersehbaren Schaden begrenzt, soweit nicht Vorsatz oder grobe Fahrlässigkeit vorliegen oder wegen der Verletzung des Lebens, des Körpers oder der Gesundheit gehaftet wird. Eine Änderung der Beweislast zu Ihrem Nachteil ist mit den vorstehenden Regelungen nicht verbunden. Von in diesem Zusammenhang bestehenden oder entstehenden Ansprüchen Dritter stellen Sie Siemens frei, soweit Siemens nicht gesetzlich zwingend haftet.
Durch Nutzung der Anwendungsbeispiele erkennen Sie an, dass Siemens über die beschriebene Haftungsregelung hinaus nicht für etwaige Schäden haftbar gemacht werden kann.

Weitere Hinweise

Siemens behält sich das Recht vor, Änderungen an den Anwendungsbeispielen jederzeit ohne Ankündigung durchzuführen. Bei Abweichungen zwischen den Vorschlägen in den Anwendungsbeispielen und anderen Siemens Publikationen, wie z. B. Katalogen, hat der Inhalt der anderen Dokumentation Vorrang.
Ergänzend gelten die Siemens Nutzungsbedingungen (https://support.industry.siemens.com).

Securityhinweise

Siemens bietet Produkte und Lösungen mit Industrial Security-Funktionen an, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen.
Um Anlagen, Systeme, Maschinen und Netzwerke gegen Cyber-Bedrohungen zu sichern, ist es erforderlich, ein ganzheitliches Industrial Security-Konzept zu implementieren (und kontinuierlich aufrechtzuerhalten), das dem aktuellen Stand der Technik entspricht. Die Produkte und Lösungen von Siemens formen nur einen Bestandteil eines solchen Konzepts.
Der Kunde ist dafür verantwortlich, unbefugten Zugriff auf seine Anlagen, Systeme, Maschinen und Netzwerke zu verhindern. Systeme, Maschinen und Komponenten sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn und soweit dies notwendig ist und entsprechende Schutzmaßnahmen (z.B. Nutzung von Firewalls und Netzwerksegmentierung) ergriffen wurden.
Zusätzlich sollten die Empfehlungen von Siemens zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Industrial Security finden Sie unter: https://www.siemens.com/industrialsecurity.
Die Produkte und Lösungen von Siemens werden ständig weiterentwickelt, um sie noch sicherer zu machen. Siemens empfiehlt ausdrücklich, Aktualisierungen durchzuführen, sobald die entsprechenden Updates zur Verfügung stehen und immer nur die aktuellen Produktversionen zu verwenden. Die Verwendung veralteter oder nicht mehr unterstützter Versionen kann das Risiko von Cyber-Bedrohungen erhöhen.
Um stets über Produkt-Updates informiert zu sein, abonnieren Sie den Siemens Industrial Security RSS Feed unter: https://www.siemens.com/industrialsecurity.

Inhaltsverzeichnis

1 Einführung

1.1 Überblick

Codequalität ist eine fundamentale Säule in der Softwareentwicklung, die mit Hilfe von Standards, Richtlinien, Ratschlägen oder "Best practice" Beispielen gefördert wird. Überprüfungen, Analysen und Tests können Aufschluss über Lücken, Fehler oder Optimierungspotential geben.

Tabelle 1-1 Stufen der Spezifikationen

Spezifikation	Hauptziel	Qualität	Umsetzung, Werkzeug
Programmierstil	Verständlichkeit	Empirisch	Code Review, Style Check
Programmiertechnik	Konformität	Pragmatisch	Statische Code-Analyse, Lint
Programmiertechnik	Effizienz	Pragmatisch	Dynamische Code-Analyse, Profiling
Testfälle	Funktionalität	Syntaktisch	Funktionstest, Unit/Integration Test
Mathematisches Modell	Korrektheit, Vollständigkeit	Semantisch	Formale Verifikation, Modell Check

Mit dieser Applikation "Projekt Check" können Sie TIA Portal-Projekte automatisiert oder manuell anhand von spezifischen Regeln überprüfen.

Die Siemens AG stellt für den Projekt Check einen Regelsatz anhand des "Programmierstyleguide für SIMATIC S7-1200 / S7-1500" \3\ zur Verfügung, der bereits enthalten ist.

Die detaillierten Ergebnisse der Überprüfung können Sie in eine Excel-Datei oder XML-Datei ausleiten und auswerten.

Sie können weitere Regelsätze mit Hilfe des mitgelieferten "Projekt Check SDK" in den Programmiersprachen Visual Basic (.NET) oder C# entwickeln und bereitstellen. Damit sind auch Ihre komplexen Regeln implementierbar.

Der Projekt Check fokussiert sich auf die erste Stufe, dem Programmierstil, auch wenn diese Applikation bereits manche – aber nicht alle – Vorgaben zu Programmiertechniken überprüfen kann.

Hinweis

Seit TIA Portal V16 gibt es das **Optionspaket "TIA Portal Test Suite Advanced"**. \8\

In V16 umfasst das integrierte Produkt eine Prüfung gegen Styleguides sowie einen Applikationstest. Die Stilregeln werden tabellarisch und die Funktionstests textuell in TIA Portal Editoren erstellt.

1.2 Funktionsweise

Der Projekt Check ist eine eigenständige Windows-Applikation, die auf ein bestehendes, lokales TIA Portal-Projekt oder auch auf eine bereits geöffnete Session des TIA Portal-Projektservers zugreifen kann. Dabei werden mit Hilfe der Programmierschnittstelle "TIA Portal Openness" die Projektdaten extrahiert, aufbereitet und den Regeln zur Überprüfung übergeben. Die Regeln erzeugen Ergebnisse, die betrachtet und exportiert werden können.

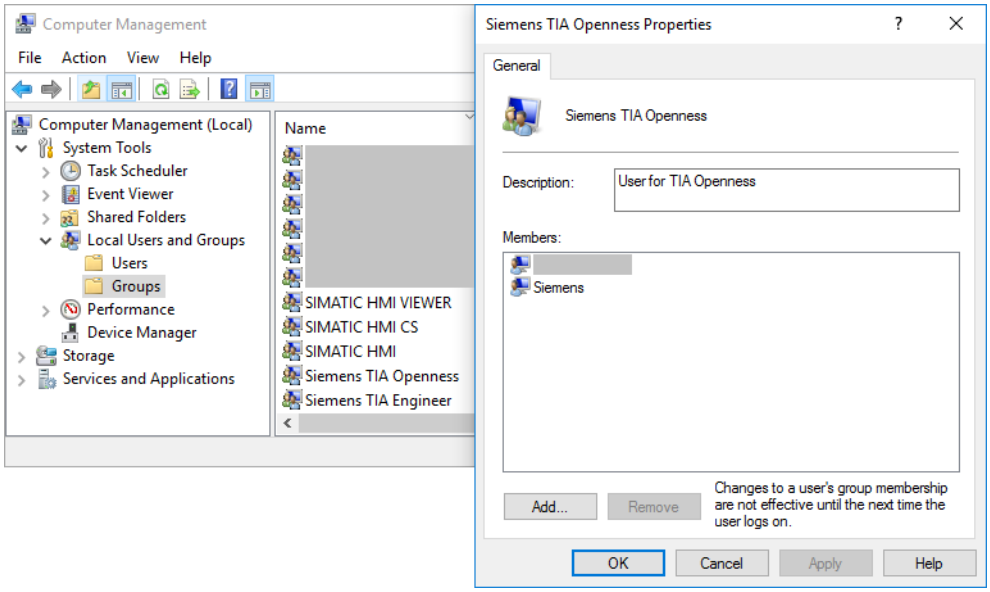
Der Projekt Check bietet zwei ausführbare Windows-Applikationen an:

1. Die Konsolenapplikation "Siemens.ProjectCheckConsole.exe" kann für automatisierte Prozesse genutzt werden, zum Beispiel für eine automatische Überprüfung im Hintergrund.
2. Die Desktopapplikation "Siemens.ProjectCheck.exe" kann für die manuelle Überprüfung genutzt werden.

Hinweis Weitere Informationen zur Verwendung der Konsolenapplikation erhalten Sie mit dem Parameter "--help".

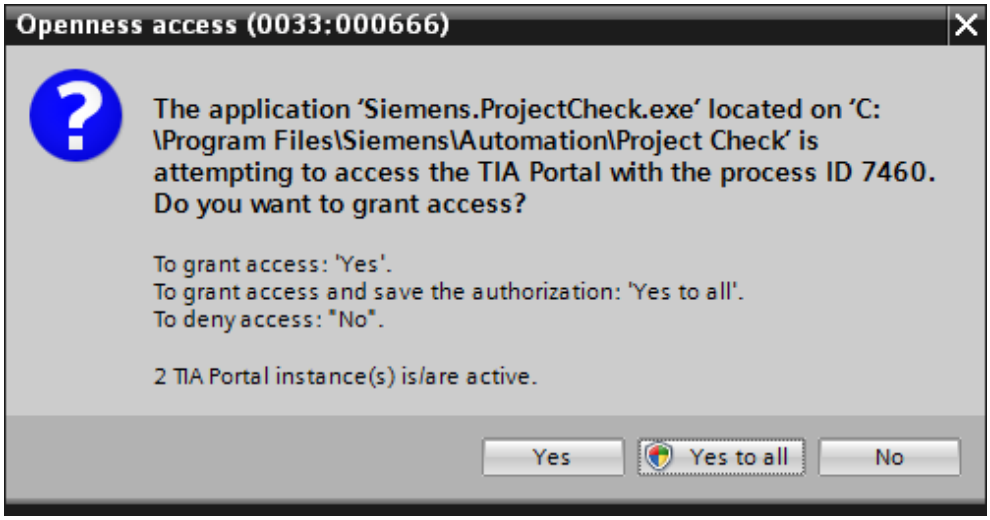
Stellen Sie vor der Verwendung des Projekt Check sicher, dass Ihr Windows-Benutzerkonto ein Mitglied der Windows-Benutzergruppe "Siemens TIA Openness" ist und dass Sie nach dem Ändern der Gruppenmitgliedschaft den Computer neu gestartet haben. Durch diese Gruppenmitgliedschaft werden Sie berechtigt, Openness-Applikationen auszuführen:

Abbildung 1-1 Computerverwaltung



Beim erstmaligen Zugriff auf TIA Portal durch den Projekt Check erscheint ein Dialog der Openness-Firewall, die Sie bitte mit "Yes to all" bestätigen. Dadurch berechtigen Sie die Applikation dauerhaft, die Projektdaten auszulesen. Wird der Dialog nur mit "Yes" bestätigt, erscheint er beim nächsten Projekt erneut:

Abbildung 1-2 Openness-Firewall



1.3 Installation

Entpacken Sie das erhaltene Archiv \2\ in einen lokalen Speicherort für Programme, zum Beispiel: "C:\Program Files\Siemens\Automation\Project Check\".

Hinweis

Aus Sicherheitsgründen wird empfohlen,

- die Applikation mit Administratorrechten in den Programm-Ordner zu installieren, um das dynamische Laden von Modulen aus dem Benutzerbereich zu vermeiden.
- die Applikation nur mit Benutzerrechten auszuführen.

Die Regelsätze werden im Unterverzeichnis "Rules" flach als Programmbibliotheken abgelegt. Andere Verzeichnisse oder Unterverzeichnisse werden von der Applikation nicht berücksichtigt.

1.4 Kompatibilität

Diese Applikation ist ab TIA Portal V14 SP1 für alle 64-Bit-Versionen verwendbar, die die langzeitstabilen Openness-Bibliotheken für V14 SP1, V15, V15.1 oder V16 mitliefern. Das können auch neuere TIA Portal-Versionen sein.

TIA Portal und das Optionspaket "TIA Portal Openness" müssen installiert und lizensiert sein.

Hinweis

Falls die Applikation die Fehlermeldung "No supported TIA Portal installation could be found" anzeigt und sowohl TIA Portal als auch das Optionspaket "TIA Portal Openness" installiert sind, führen Sie bitte eine Reparaturinstallation für TIA Portal durch.

Die Applikation setzt Microsoft .NET Framework 4.8 Runtime voraus. \6\, \7\

1.5 Systemeigenschaften

Die Applikation wurde unter folgender Konfiguration getestet:

Tabelle 1-2 Testkonfiguration

Produkt	Version
Microsoft Windows	10 Version 1607 (64-Bit-Architektur)
Microsoft Office	2016
Microsoft .NET Framework Runtime	4.8
TIA Portal	• V14 SP1 Update 9 • V15 Update 4 • V15.1 Update 4 • V16 Update 2
Installierte TIA Portal-Software	STEP7 Professional
Installierte TIA Portal-Optionen	TIA Portal Openness

Hinweis

Microsoft Office muss nicht auf dem System installiert sein. Sie können die Ergebnisse dennoch exportieren und die erzeugte Datei auf einem anderen Computer öffnen.

2 Handhabung

2.1 Desktopapplikation

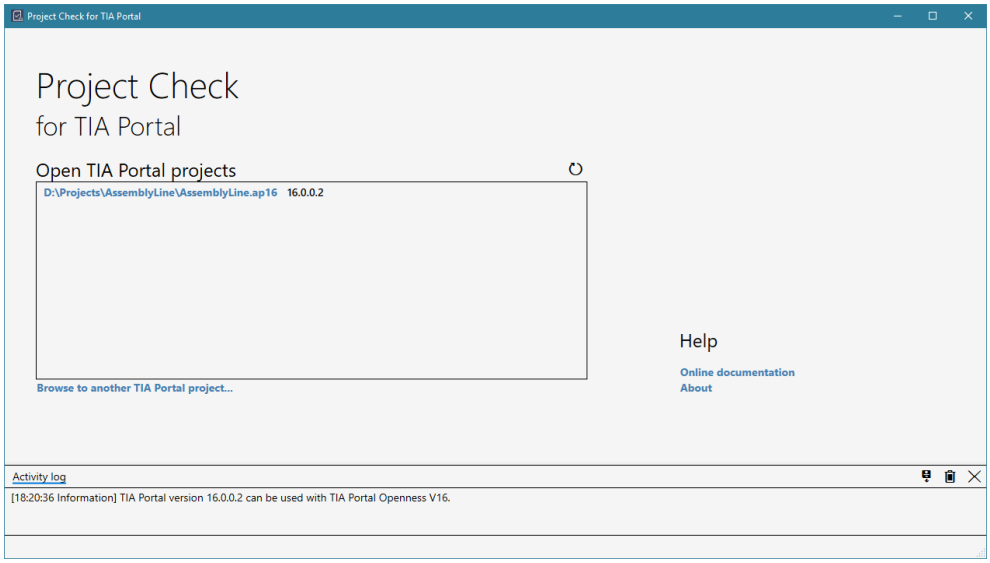
Starten Sie die Desktopapplikation "Siemens.ProjectCheck.exe" per Doppelklick.

Allgemein

Die Applikation lässt sich durch das "X" am Fenster rechts oben beenden. Die Applikation lässt sich nur beenden, wenn keine Aktion ausgeführt wird.

Im unteren Bereich des Fensters befindet sich das "Activity log", das Sie jederzeit durch einen Klick einblenden können. Dort können Sie die Aktionen mitverfolgen. Sie können dieses Protokoll speichern, leeren und ausblenden.

Abbildung 2-1 "Activity log"

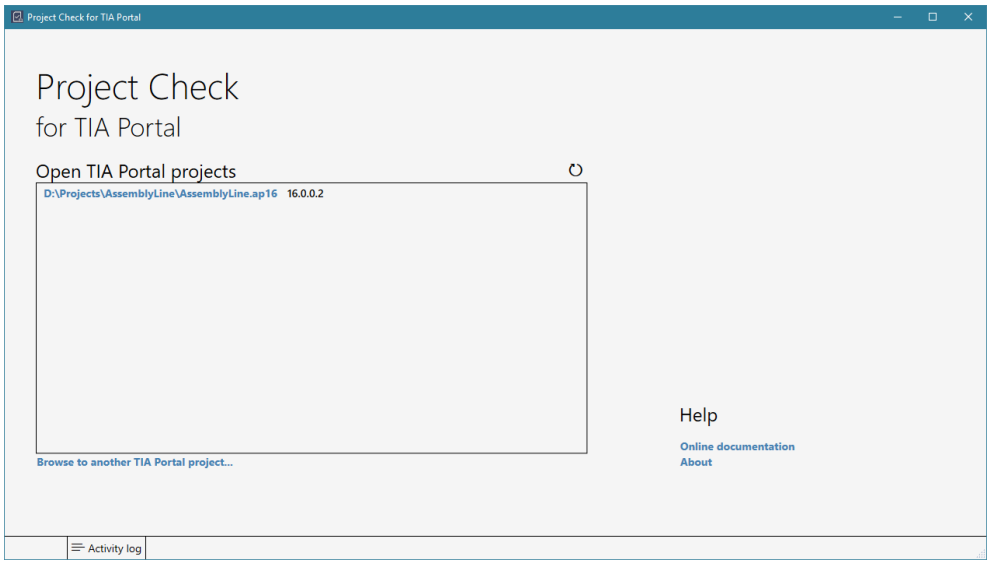


Startseite

Auf der Startseite sehen Sie eine Liste der bereits offenen TIA Portal-Projekte. Sie können diese Liste aktualisieren und einen Eintrag daraus anklicken, um das Projekt zu prüfen. Alternativ können Sie über die Schaltfläche "Browse to another TIA Portal project…" ein anderes TIA Portal-Projekt öffnen lassen.

Im rechten, unteren Bereich können Sie die Hilfethemen aufrufen.

Abbildung 2-2 Startseite



Projektseite

Nach dem Auswählen oder Öffnen eines Projekts wird auf der Projektseite im linken Bereich ein Projektbaum angezeigt. Sie können den Projektbaum aktualisieren lassen, falls Sie in der Zwischenzeit Änderungen im TIA Portal-Projekt gemacht haben. Außerdem können Sie den Projektbaum komplett auf- und zuklappen.

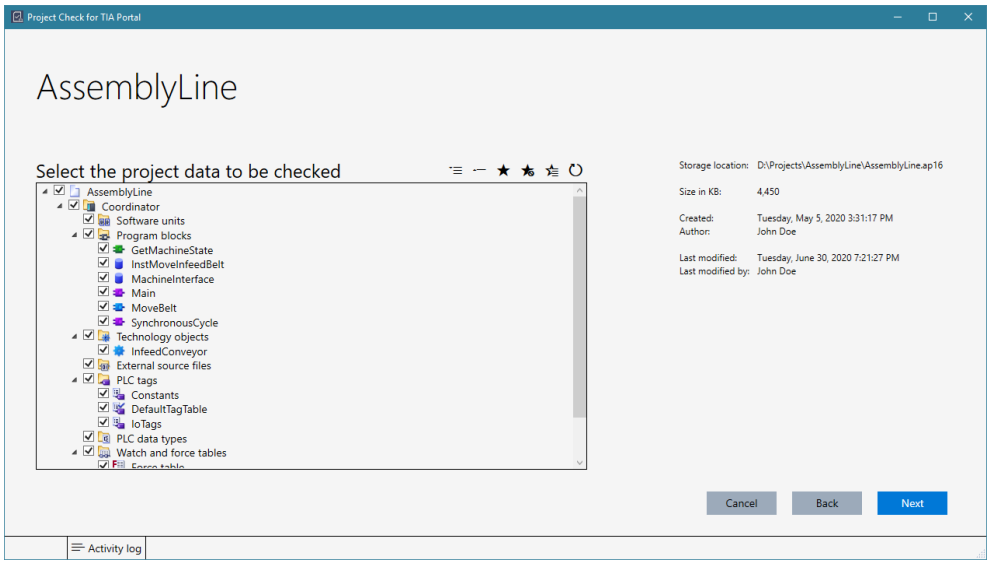
Im rechten Bereich sehen Sie allgemeine Projektinformationen.

Im Projektbaum können Sie über die Auswahlboxen den zu überprüfenden Inhalt festlegen. Diese Auswahl (Einschlussliste oder Ausschlussliste) können Sie sich über die Schaltflächen in einer separaten Datei speichern und für eine wiederholte Prüfung laden.

Über die Schaltflächen "Cancel" und "Back" gelangen Sie zurück zur Startseite. Dabei werden die Verbindung zu TIA Portal getrennt und die Projektdaten zurückgesetzt.

Über die Schaltfläche "Next" gelangen Sie zur Regelseite. Die Schaltfläche wird aktiv, sobald Sie mindestens ein Objekt aus dem Projektbaum angehakt haben.

Abbildung 2-3 Projektseite



Regelseite

Nach dem Festlegen des zu prüfenden Projektinhalts wird auf der Regelseite ein Regelbaum angezeigt. Dort sehen Sie alle geladenen Regelsätze mit ihren Kategorien und Regeln. Sie können den Regelbaum komplett auf- und zuklappen.

Legen Sie die anzuwendenden Regeln über die Auswahlboxen fest. Diese Auswahl (Einschlussliste oder Ausschlussliste) können Sie sich über die Schaltflächen in einer separaten Datei speichern und für eine wiederholte Prüfung laden.

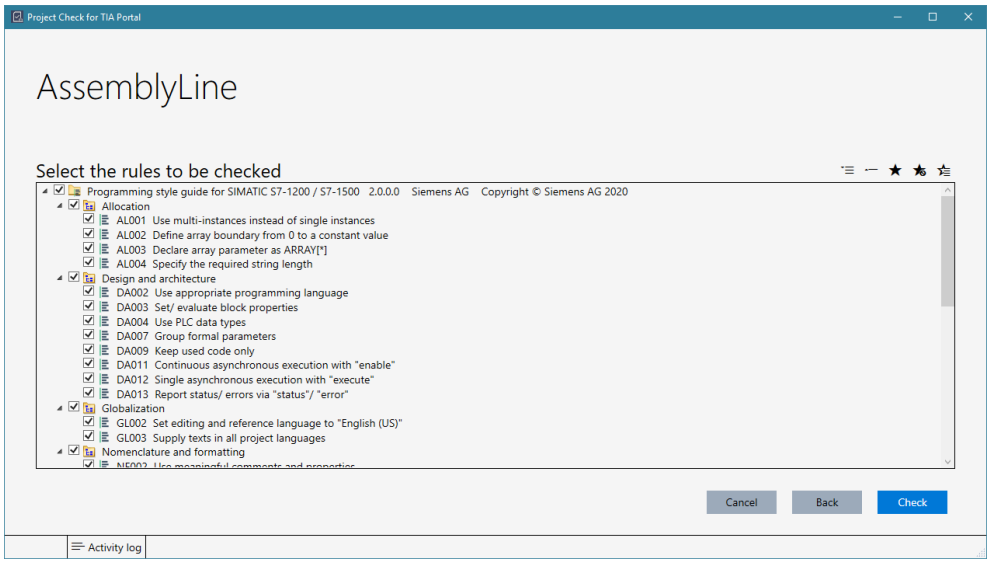
Über die Schaltfläche "Cancel" gelangen Sie zurück zur Startseite. Dabei werden die Verbindung zu TIA Portal getrennt sowie die Projektdaten und die Regelauswahl zurückgesetzt.

Über die Schaltfläche "Back" gelangen Sie zurück zur Projektseite. Dabei wird die Regelauswahl zurückgesetzt.

Über die Schaltfläche "Check" wird die Überprüfung gestartet. Die Schaltfläche wird aktiv, sobald Sie mindestens eine Regel aus dem Regelbaum angehakt haben. Die Überprüfung kann je nach Umfang des Projekts und der Regeln mehrere Minuten dauern. Anschließend gelangen Sie zur Ergebnisseite.

Hinweis Die Überprüfung beginnt mit der Datenextraktion aus dem TIA Portal-Projekt. Währenddessen ist das TIA Portal durch den Projekt Check exklusiv verriegelt.

Abbildung 2-4 Regelseite



Ergebnisseite

Nach Abschluss der Überprüfung werden auf der Ergebnisseite im oberen Bereich die Anzahl der Fehler, Warnungen und Informationen angezeigt. Darunter sehen Sie im Reiter "Results" alle Ergebnisse, die Sie filtern können.

Per Klick auf den grünen Pfeil eines Ergebnisses können Sie zum entsprechenden Editor in TIA Portal springen.

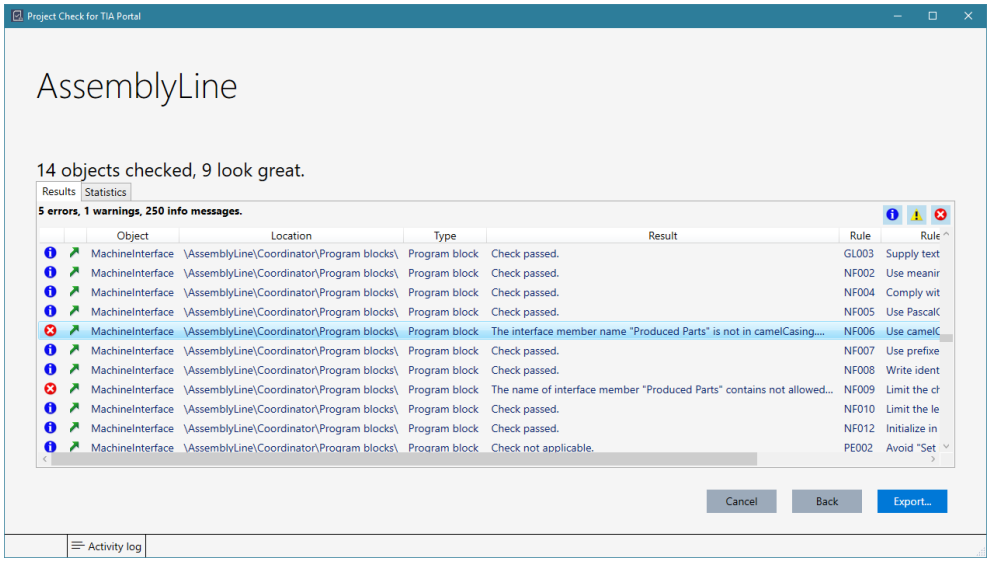
Im Reiter "Statistics" sehen Sie die zehn Objekte mit den meisten Fehlern und Warnungen sowie die zehn Regeln mit den meisten Fehlern und Warnungen.

Über die Schaltfläche "Cancel" gelangen Sie zurück zur Startseite. Dabei werden die Verbindung zu TIA Portal getrennt sowie die Projektdaten, Regelauswahl und Ergebnisse zurückgesetzt.

Über die Schaltfläche "Back" gelangen Sie zurück zur Regelseite. Dabei werden die Ergebnisse zurückgesetzt.

Über die Schaltfläche "Export" können Sie die Ergebnisse und Statistiken in eine Excel-Datei mit mehreren Arbeitsblättern oder in eine XML-Datei zur Weiterverarbeitung ausleiten. Die Generierung kann je nach Umfang mehrere Sekunden dauern.

Abbildung 2-5 Ergebnisseite



Hinweis

Die Excel-Datei enthält drei Arbeitsblätter: "Results", "Top 10 objects" und "Top 10 rules". Die XML-Datei enthält nur "Results".

Das Format können Sie im "Speichern"-Dialog auswählen.

Es werden alle Ergebnisse exportiert, unabhängig vom gewählten Filter.

2.2 Konsolenapplikation

Starten Sie die Konsolenapplikation "Siemens.ProjectCheckConsole.exe" in der Windows-Eingabeaufforderung oder in der Windows-PowerShell mit den folgenden Parametern:

Tabelle 2-1 Parameter der Konsolenapplikation

Kurzform	Langform	Erforderlich	Beschreibung
`-q`	`--quiet`	Nein	Ausgabe des "Activity log" in der Konsole ausschalten
`-l`	`--logfile`	Nein	Pfad für die Protokolldatei
`-p`	`--project`	**Ja**	Pfad zur TIA Portal-Projektdatei
`-r`	`--report`	**Ja**	Pfad(e) für die Ergebnisdatei(en) (*.xml oder *.xlsx)
`-s`	`--filter-rules`	Nein	Pfad zur Regelfilter-Datei
`-o`	`--filter-project`	Nein	Pfad zur Projektfilter-Datei
n. v.	`--help`	Nein	Hilfe anzeigen
n. v.	`--version`	Nein	Version anzeigen

Ein Aufruf in der Windows-PowerShell kann zum Beispiel wie folgt aussehen. Der Aufruf passiert in einer einzigen Zeile:

```
.\Siemens.ProjectCheckConsole.exe
 -p "D:\Projects\Project1\Project1.ap16"
 -l "D:\Projects\Project1\UserFiles\ProjectCheck.log"
 -r "D:\Projects\Project1\UserFiles\ProjectCheckReport.xml"
    "D:\Projects\Project1\UserFiles\ProjectCheckReport.xlsx"
```

Die Konsolenapplikation nutzt die folgenden Exit-Codes:

Tabelle 2-2 Exit-Codes der Konsolenapplikation

Exit-Code	Bedeutung
`0`	Die Anwendung wurde ohne Fehler beendet.
`1`	Die Anwendung wurde mit einem Fehler beendet.

"Fehler" meint in diesem Zusammenhang nicht, ob bei der Überprüfung in einem TIA Portal Projekt Fehler gefunden wurden.

Für eine Auswertung der Projektfehler analysieren Sie bitte die Ergebnisse in der exportierten Datei.

Hinweis

Die Excel-Datei enthält drei Arbeitsblätter: "Results", "Top 10 objects" und "Top 10 rules". Die XML-Datei enthält nur "Results".

Es werden alle Ergebnisse exportiert.

2.3 Umfang

Der Projekt Check kann folgende Objekte eines TIA Portal-Projekts überprüfen. Jedes Objekt stellt gewisse Eigenschaften bereit, die im Projekt Check SDK beschrieben sind.

- Projekt
- Gruppe / Ordner
- PLC (ab V16 mit Eigenschaft "Simulierbarkeit")
- Software-Unit (ab V16 verfügbar)
- Programm-Baustein (ohne Inhalt für ProDiag; ab V15 auch mit SimaticML für SCL-Baustein und auch KOP/FUP-Baustein, der SCL-Netzwerke enthält)
  - Element einer Bausteinschnittstelle (ab V15 auch für KOP/FUP-Baustein, der SCL-Netzwerke enthält)
  - Kindelement eines Elements (nur für Struct und Array von Struct)
- Technologie-Objekt (ab V15 mit Editor öffnen, ab V16 mit SimaticML für aktuelle TO-Versionen)
- Externe Quelle
- PLC-Variablentabelle
  - PLC-Variable
  - PLC-Anwenderkonstante
- PLC-Datentyp
  - Element eines PLC-Datentyps
  - Kindelement eines Elements (nur für Struct und Array von Struct)
- Force-Tabelle (ab V15.1 verfügbar)
  - Eintrag einer Force-Tabelle (ab V15.1 verfügbar)
- Beobachtungstabelle (ab V15.1 verfügbar)
  - Eintrag einer Beobachtungstabelle (ab V15.1 verfügbar)
- Kopiervorlage (ohne Inhalt)
- Bibliothekstyp eines Programbausteins
  - Version eines Programmbausteins (ohne Inhalt)
- Bibliothekstyp eines PLC-Datentyps
  - Version eines PLC-Datentyps (ohne Inhalt)

Hinweis

Der Umfang ist durch den Funktionsumfang der Programmierschnittstelle "TIA Portal Openness" in der jeweiligen TIA Portal-Version limitiert.

Systembausteine, Systemdatentypen und Systemkonstanten werden übersprungen.

Systemelemente in einer Bausteinschnittstelle stehen zur Verfügung und sind mit "IsReadOnly" markiert.

2.4 Hinweise und Fehlerbehandlung

Halten Sie Ihre TIA Portal-Installation auf dem neusten Stand, in dem Sie alle Updates installieren.

In der Desktopapplikation können Sie das "Activity log" nutzen, um die Aktionen mitzuverfolgen.

In der Konsolenapplikation können Sie das "Activity log" direkt ausgeben lassen oder in einer separaten Datei protokollieren lassen.

Der Projekt Check kann nur auf Projekte in TIA Portal zugreifen, wenn alle für das Projekt benötigten TIA Portal-Pakete, Hardware Support Packages und GSD-Dateien installiert und lizensiert sind.

Es müssen alle Dialogfenster in TIA Portal geschlossen sein, damit der Projekt Check auf die Daten zugreifen kann. TIA Portal-Editoren sind davon nicht betroffen.

Kompilieren Sie alle PLCs im TIA Portal-Projekt, bevor Sie die Überprüfung starten, um eine vollständige Überprüfung durchzuführen. Nur konsistente Daten können vollständig überprüft werden.

Der Projekt Check unterstützt TIA Portal-Projekte mit PLCs der S7-1200 / S7-1500 Familie. Andere PLCs werden übersprungen.

Bei Know-how geschützten Programmbausteinen kann nur die Bausteinschnittstelle ausgelesen werden.

Die Baumansicht des TIA Portal-Projekts ist an TIA Portal angelehnt, zeigt aber nur wesentliche Informationen an. Beispielsweise werden Informationen zu Know-how-Schutz und Versionierung nicht dargestellt. Auch die Sortierung kann abweichen.

Der Knoten "Software units" wird nur bei unterstützten PLCs angezeigt.

3 Regelsatz "Programmierstyleguide für SIMATIC S7-1200 / S7-1500"

Die Regeln und Empfehlungen aus dem "Programmierstyleguide für SIMATIC S7 1200 / S7-1500" \3\ wurden in einen gleichnamigen Regelsatz implementiert:

Tabelle 3-1 Implementierter Regelsatz

Eigenschaft	Wert
Implementierte Version	V2.0
Programmbibliothek für den Regelsatz	Siemens.ProgrammingStyleGuide.dll

Eine Verletzung einer Empfehlung wird als "Warnung" dargestellt, eine Verletzung einer Regel als "Fehler", alle bestandenen oder nicht anwendbaren Prüfungen als "Info".

Die folgenden Tabellen geben eine Übersicht über den Implementierungsumfang:

Tabelle 3-2 Einstellungen in TIA Portal

Nr.	Beschreibung	Bemerkung
ES001	Oberflächensprache "English"	Nicht implementiert. Hierbei handelt es sich um Regeln und Empfehlungen für TIA Portal-Einstellungen. Auf diese besteht kein Zugriff. Die Einstellungen sind unabhängig vom TIA Portal-Projekt.
ES002	Mnemonik "International"	
ES003	Nichtproportionale Schriftart für Editoren	
ES004	Smarte Einrückung mit zwei Leerzeichen	
ES005	Symbolische Repräsentation von Operanden	
ES006	IEC-konforme Programmierung	
ES007	Expliziter Datenzugriff per HMI/ OPC UA/ Web API	
ES008	Automatische Wertprüfung (ENO) aktiviert	
ES009	Automatische Prüfung von Arraygrenzen	

Tabelle 3-3 Globalisierung

Nr.	Beschreibung	Bemerkung
GL001	Einheitliche Sprache verwenden	Nicht implementiert. Es benötigt Grammatik-Analyse.
GL002	Editier- und Referenzsprache "English (US)" setzen	Implementiert.
GL003	Texte in allen Projektsprachen hinterlegen	Teilweise implementiert. Texte im Code werden nicht geprüft, da es Code-Analyse benötigt.

Tabelle 3-4 Nomenklatur und Formatierung

Nr.	Beschreibung	Bemerkung
NF001	Eindeutig und einheitlich in Englisch bezeichnen	Nicht implementiert. Es benötigt Grammatik-Analyse.
NF002	Sinnvolle Kommentare & Eigenschaften verwenden	Teilweise implementiert. Die "Sinnhaftigkeit" kann nicht geprüft werden.
NF003	Entwicklerinformationen dokumentieren	Nicht implementiert. Es gibt kein festes Format.

Nr.	Beschreibung	Bemerkung
NF004	Präfixe und Struktur für Bibliotheken einhalten	Implementiert.
NF005	Objekte in PascalCasing bezeichnen	Implementiert. Die Prüfung erfolgt ohne Wörterbuch.
NF006	Codeelemente in camelCasing bezeichnen	Implementiert. Die Prüfung erfolgt ohne Wörterbuch.
NF007	Präfixe verwenden	Implementiert.
NF008	Bezeichner von Konstanten GROSS schreiben	Implementiert.
NF009	Zeichensatz für Bezeichner einschränken	Implementiert.
NF010	Zeichenlänge für Bezeichner einschränken	Implementiert.
NF011	Nur eine Abkürzung pro Bezeichner nutzen	Nicht implementiert. Es benötigt Grammatik-Analyse.
NF012	Im konformen Format initialisieren	Implementiert.
NF013	Optionale Formalparameter ausblenden	Nicht implementiert. Auf diese Einstellung besteht kein Zugriff.
NF014	SCL-Code sinnvoll formatieren	Nicht implementiert. Es benötigt Code-Analyse.

Tabelle 3-5 Wiederverwendbarkeit

Nr.	Beschreibung	Bemerkung
RU001	Simulierbare Bausteine bereitstellen	Implementiert.
RU002	Vollständig mit Bibliotheken versionieren	Implementiert.
RU003	Nur freigegebene Typen in fertigen Projekten halten	Implementiert.
RU004	Nur lokale Variablen verwenden	Implementiert.
RU005	Lokale symbolische Konstanten verwenden	Teilweise implementiert. Die Verwendung von globalen Konstanten oder "Magic Numbers" werden nicht geprüft, da es Code-Analyse benötigt.
RU006	Vollsymbolisch programmieren	Implementiert.
RU007	Hardwareunabhängig programmieren	Teilweise implementiert. Die Verwendung von Systemfunktionen wird nicht geprüft, da es Code-Analyse benötigt.
RU008	Vorlagen verwenden	Nicht implementiert. Es gibt verschiedene Vorlagen, die unterschiedlich oder nur teilweise genutzt werden können.

Tabelle 3-6 Referenzieren von Objekten (Allokieren)

Nr.	Beschreibung	Bemerkung
AL001	Multiinstanzen statt Einzelinstanzen nutzen	Implementiert.
AL002	Arraygrenze von 0 bis Konstante definieren	Implementiert.
AL003	Array-Parameter als ARRAY[*] deklarieren	Implementiert.

Nr.	Beschreibung	Bemerkung
AL004	Benötigte String-Länge festlegen	Implementiert.

Tabelle 3-7 Sicherheit

Nr.	Beschreibung	Bemerkung
SE001	Aktualwerte auf Gültigkeit prüfen	Nicht implementiert. Es benötigt Code-Analyse.
SE002	Temporäre Variablen initialisieren	Nicht implementiert. Es benötigt Code-Analyse.
SE003	ENO behandeln	Teilweise implementiert: Es wird die Eigenschaft "ENO automatisch setzen" geprüft. Code wird nicht geprüft, da es Code-Analyse benötigt.
SE004	Datenzugriff per HMI/ OPC UA/ Web API selektiv aktivieren	Implementiert. Die gewünschte "Selektivität" kann nicht hinterlegt werden.
SE005	Fehlercodes auswerten	Nicht implementiert. Es benötigt Code-Analyse.
SE006	Fehler-OB mit Auswertelogik schreiben	Nicht implementiert. Es benötigt Code-Analyse.

Tabelle 3-8 Designrichtlinien / Architektur

Nr.	Beschreibung	Bemerkung
DA001	Projekt/ Bibliothek gruppieren und strukturieren	Nicht implementiert. Es gibt kein Merkmal für "logische Einheiten".
DA002	Geeignete Programmiersprache verwenden	Implementiert.
DA003	Bausteineigenschaften setzen/ prüfen	Implementiert.
DA004	PLC-Datentypen verwenden	Implementiert. "Struct" in Lokaldaten wird ignoriert, dessen Verwendung wird nicht geprüft.
DA005	Daten nur über Formalparameter austauschen	Nicht implementiert. Es benötigt Code-Analyse.
DA006	Auf statische Variablen nur lokal zugreifen	Nicht implementiert. Es benötigt Code-Analyse.
DA007	Formalparameter zusammenfassen	Implementiert.
DA008	Ausgangsparameter genau einmal schreiben	Nicht implementiert. Es benötigt Code-Analyse.
DA009	Nur genutzten Code beibehalten	Teilweise implementiert. Es benötigt Code-Analyse. Nur die Existenz von externen Quellen wird geprüft.
DA010	Asynchrone Bausteine nach PLCopen entwickeln	Nicht implementiert. Die Regel ist nur eine allgemeine Vorgabe.
DA011	Kontinuierliche asynchrone Abarbeitung mit "enable"	Teilweise implementiert. Es werden nur Bausteine geprüft, die einen Eingang "enable" haben.
DA012	Einmalige asynchrone Abarbeitung mit "execute"	Teilweise implementiert. Es werden nur Bausteine geprüft, die einen Eingang "execute"

Nr.	Beschreibung	Bemerkung
		haben.
DA013	Status/ Fehler per "status"/ "error" zurückgeben	Teilweise implementiert. Es werden nur Bausteine geprüft, die einen Ausgang "error" oder "status" haben.
DA014	Standardisierte Wertebereiche in "status" verwenden	Nicht implementiert. Es gibt kein Merkmal für "Korrektheit".
DA015	Unterlagerte Informationen durchreichen	Nicht implementiert. Die Diagnosestruktur ist nicht fest vorgeschrieben.
DA016	CASE-Anweisung statt ELSIF-Zweige nutzen	Nicht implementiert. Es benötigt Code-Analyse.
DA017	ELSE-Zweig bei CASE-Anweisungen erstellen	Nicht implementiert. Es benötigt Code-Analyse.
DA018	Jump und Label vermeiden	Nicht implementiert. Es benötigt Code-Analyse.

Tabelle 3-9 Performance

Nr.	Beschreibung	Bemerkung
PE001	"Create extended status info" deaktivieren	Nicht implementiert. Auf diese Einstellung besteht kein Zugriff.
PE002	"Set in IDB" vermeiden	Implementiert.
PE003	Strukturierte Parameter als Referenz übergeben	Implementiert.
PE004	Formalparameter mit Variant vermeiden	Implementiert.
PE005	Formalparameter "mode" vermeiden	Implementiert.
PE006	Temporäre Variablen bevorzugen	Nicht implementiert. Es benötigt Code-Analyse.
PE007	Wichtige Testvariablen statisch deklarieren	Nicht implementiert: Es benötigt Code-Analyse. Es gibt kein Merkmal für "Wichtigkeit".
PE008	Lauf-/ Index-Variablen als "DInt" deklarieren	Nicht implementiert. Es benötigt Code-Analyse.
PE009	Mehrmaligen, gleichen Indexzugriff vermeiden	Nicht implementiert. Es benötigt Code-Analyse.
PE010	Slice anstelle von Maskierungen verwenden	Nicht implementiert. Es benötigt Code-Analyse.
PE011	IF/ ELSE-Anweisungen vereinfachen	Nicht implementiert. Es benötigt Code-Analyse.
PE012	IF/ ELSIF-Fälle nach Erwartung sortieren	Nicht implementiert. Es benötigt Code-Analyse. Es gibt kein Merkmal für "Erwartung".
PE013	Speicherintensive Anweisungen vermeiden	Nicht implementiert. Es benötigt Code-Analyse.
PE014	Laufzeitintensive Anweisungen vermeiden	Nicht implementiert. Es benötigt Code-Analyse.
PE015	SCL/ KOP/ FUP für zeitkritische Anwendungen nutzen	Implementiert.
PE016	Einstellung der Mindestzykluszeit prüfen	Implementiert.

4 Projekt Check SDK

4.1 Überblick

SDK bedeutet "Software Development Kit". Das Projekt Check SDK stellt eine Programmbibliothek bereit, die zur Entwicklung eigener Regelsätze dient.

Hinweis Erfahrung in der Entwicklung von Microsoft Windows-Programmbibliotheken mit dem Microsoft .NET Framework wird vorausgesetzt.

Das SDK ist ein Bestandteil des Applikationsbeispiels. Die SDK-Dateien finden Sie im gleichen Ordner wie die ausführbaren Applikationen.

Tabelle 4-1 Eigenschaften des Projekt Check SDK

Eigenschaft	Wert
Name	Siemens.Applications.TiaPortal.ProjectCheck.Sdk.dll
.NET Version	4.8
Assembly Dateiversion	3.0.0
Assembly Version	3.0.0
Public Token	1208d84b9f643f11
XML-Dokumentation (Intellisense)	Siemens.Applications.TiaPortal.ProjectCheck.Sdk.xml
PDF-Dokumentation (Schnittstellen, Klassen)	Siemens.Applications.TiaPortal.ProjectCheck.Sdk.pdf

4.2 Umfang

Siehe auch Kapitel 2.3.

Grundsätzlich stellt das Projekt Check SDK die Daten nur lesend zur Verfügung. Des Weiteren werden für konsistente Objekte in vielen Klasse die Eigenschaften `SimaticMl` und `Source` angeboten, falls für eine eigene Regel die aufbereiteten Eigenschaften nicht ausreichen.

4.3 Verwendung

Ein Regelsatz wird als Programmbibliothek bereitgestellt. Mithilfe des Projekt Check SDK können Sie eine solche Programmbibliothek entwickeln.

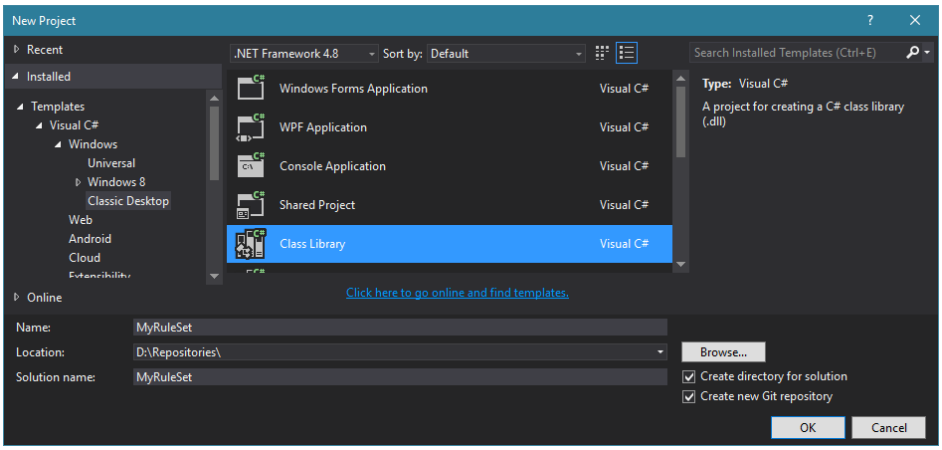
Hinweis

- In einer Programmbibliothek bilden Sie genau einen Regelsatz ab.
- Eine Programmbibliothek darf maximal 10 MiB groß sein.
- Ein Regelsatz kann aus mehreren Regeln bestehen.
- Eine Regel kann mehrere unterschiedliche Objekte überprüfen.

Dazu gehen Sie wie folgt vor:

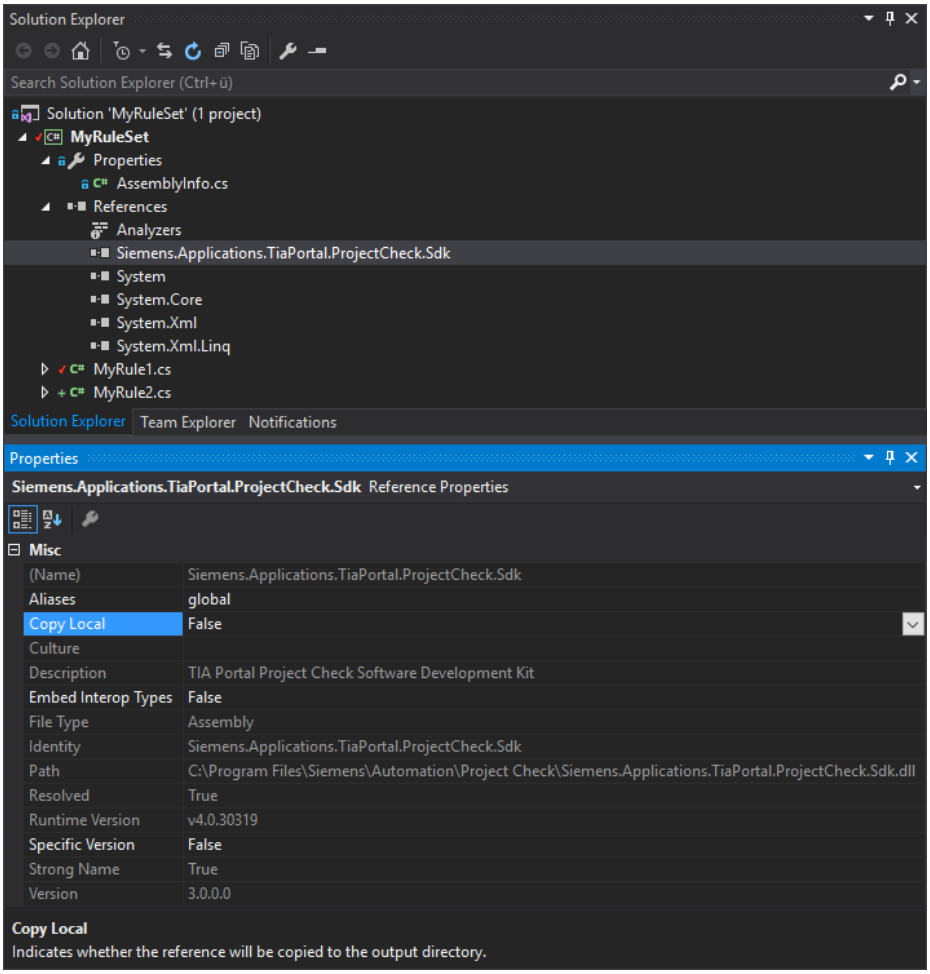
1. Starten Sie Microsoft Visual Studio (Version 2015 Update 3 oder neuer).
2. Erstellen Sie ein neues Projekt vom Typ "Class Library" aus der Kategorie "Visual C# > Windows > Classic Desktop" und stellen Sie die .NET Version auf ".NET Framework 4.8" ein.

Abbildung 4-1 Neues Visual Studio Projekt anlegen



3. Fügen Sie Ihrem Projekt eine Referenz zum Projekt Check SDK hinzu, das sich in Ihrem Installationsverzeichnis befindet. Stellen Sie die Eigenschaft "Copy Local" auf "False", da es ist nicht erforderlich ist, das SDK zu kopieren.

Abbildung 4-2 Referenz zum SDK



4. Öffnen Sie unter "Properties" die Datei "AssemblyInfo.cs" und stellen Sie mindestens folgende Attribute bereit:
```
[assembly: AssemblyDescription("My rule set")]
[assembly: AssemblyCompany("My Company")]
[assembly: AssemblyCopyright("Copyright © My Company 2020")]
[assembly: AssemblyFileVersion("0.1.0")]
[assembly: AssemblyVersion("0.1.0")]
```
Diese Attribute werden dem Endanwender in der Oberfläche des Projekt Check als Information zum Regelsatz angezeigt.

5. Stellen Sie zusätzlich folgende Attribute in der Datei "AssemblyInfo.cs" bereit, um die erforderliche Codezugriffssicherheit (CAS) festzulegen:
Entweder nur das Attribut
```
[assembly: SecurityTransparent]
```
oder die folgenden beiden Attribute
```
[assembly: AllowPartiallyTrustedCallers]
[assembly: SecurityRules(SecurityRuleSet.Level2)]
```
Technische Details dazu finden Sie bei Microsoft Docs unter "Sicherheitstransparenter Code, Ebene 2" \4\.

6. Legen Sie eine neue Klasse an, die Sie als `public sealed` kennzeichnen. Folgende Namespaces können Sie einbinden:
```
using Siemens.Applications.TiaPortal.ProjectCheck.Sdk.Enums;
using Siemens.Applications.TiaPortal.ProjectCheck.Sdk.Interfaces;
using Siemens.Applications.TiaPortal.ProjectCheck.Sdk.Interfaces.User;
using Siemens.Applications.TiaPortal.ProjectCheck.Sdk.Models;
using Siemens.Applications.TiaPortal.ProjectCheck.Sdk.Models.User;
using Siemens.Applications.TiaPortal.ProjectCheck.Sdk.Rules;
```

7. Vererben Sie die generische, abstrakte Klasse `BaseRule<T>` an Ihre Klasse. Dabei muss `T` Ihre eigene, gleiche Klasse sein. Implementieren Sie den erforderlichen parameterlosen Konstruktor. Dabei müssen Sie den `base`-Konstruktor aufrufen:
```
public sealed class MyRule : BaseRule<MyRule>
{
    public MyRule() : base("MR901", "My Rule", "My Category")
    {
    }
}
```

8. Überschreiben Sie alle `Check()`-Methoden für die Objekte, die Sie in Ihrer Regel überprüfen möchten. Löschen Sie den Aufruf von `base.Check(item)`, falls dieser automatisch angelegt wurde:
```
    public override void Check(Project item)
    {
        // TODO: Add check logic here
    }
```

9. Fügen Sie in jeder überschriebenen `Check()`-Methode mindestens ein Ergebnis der Sammlung hinzu. Die `Results`-Sammlung wird durch die Basisklasse bereitgestellt:
```
        Results.Add(Level.Info, item, DefaultMessage.Passed);
```

10. Fügen Sie für jede weitere Regel eine eigene Klasse hinzu.

11. Kompilieren Sie den Regelsatz und kopieren Sie die Programmbibliothek in das Unterverzeichnis "Rules" des Projekt Check.

4.4 Debuggen

Voraussetzung ist, dass Sie bereits einen eigenen Regelsatz entwickelt und dessen Programmbibliothek in das Unterverzeichnis "Rules" des Projekt Check eingefügt haben.

1. Öffnen Sie Ihr Regelsatz-Projekt in Microsoft Visual Studio.
2. Setzen Sie an gewünschten Stellen in Ihrem Regelsatz-Projekt die Haltepunkte.
3. Starten Sie die Desktopapplikation des Projekt Check.
4. Fügen Sie über Visual Studio > Debug > "Attach to Process…" den Debugger zum Projekt Check-Prozess hinzu.
5. Wählen Sie im Projekt Check ein TIA Portal-Projekt und den zu prüfenden Projektinhalt aus.
6. Beim Klick auf "Next" werden die Programmbibliotheken der Regelsätze geladen und die Konstruktoren aufgerufen.
7. Beim Klick auf "Check" werden die `Check()`-Methoden der Regeln aufgerufen.
8. Falls Sie die Programmbibliothek austauschen, müssen Sie die Applikation neu starten.

Hinweis

`Debugger.Launch()` steht nicht zur Verfügung, da dieser Aufruf kritische Berechtigungen zum Ausführen von "unmanaged code" benötigt.

Beim Debuggen ist es nicht möglich, direkt den Parameter `item` zu inspizieren. Sie werden stattdessen je nach Visual Studio-Einstellung einen dieser Werte sehen:

- `Obtaining the runtime type of a transparent proxy is not supported in this context.`
- `{System.Runtime.Remoting.Proxies.__TransparentProxy}`

4.5 Beispielregel

Anforderungen an dieses Beispiel

- Der Name aller Elemente der Bausteinschnittstelle muss kürzer als 24 Zeichen sein.
- Im Gut-Fall wird für einen Programmbaustein nur eine einzige **Information** "Check passed." angelegt.
- Ist die Bausteinschnittstelle nicht verfügbar, wird für einen Programmbaustein eine **Warnung** angelegt.
- Für jeden Verstoß in der Bausteinschnittstelle wird ein **Fehler** für den Programmbaustein angelegt.

Implementierung des Beispiels

```
using System.Collections.Generic;
using System.Linq;
using Siemens.Applications.TiaPortal.ProjectCheck.Sdk.Models;
using Siemens.Applications.TiaPortal.ProjectCheck.Sdk.Models.User;
using Siemens.Applications.TiaPortal.ProjectCheck.Sdk.Rules;
namespace MyRuleSet
{
  public sealed class MyRule : BaseRule<MyRule>
  {
    public MyRule() : base("MR901", "My Rule", "My Category")
    {
    }
    public override void Check(ProgramBlock item)
    {
      if (item.InterfaceMembers == null)
      {
        // The program block may be not consistent,
        // so the block interface may not be available.
        Results.Add(Level.Warning, item, "The block interface is not available.");
      }
      else if (CheckInterfaceMembers(item, item.InterfaceMembers.Where(e =>
               e.IsReadOnly != true)))
      {
        // All interface members have passed the check.
        Results.Add(Level.Info, item, DefaultMessage.Passed);
      }
    }
    private bool CheckInterfaceMembers(ProgramBlock item,
      IEnumerable<ProgramBlockInterfaceMember> members)
    {
      var isSuccessful = true;
      foreach (var member in members)
      {
        // Check the name of the current interface member
        if (member.Name.Length > 24)
        {
          isSuccessful = false;
          Results.Add(Level.Error, item,
            $"The name of the interface member {member.Name} " +
            $"is too long. It has {member.Name.Length} characters.");
        }
        // Check also all child interface members
        if (member.Members != null && !CheckInterfaceMembers(item, member.Members))
        {
          isSuccessful = false;
        }
      }
      return isSuccessful;
    }
  }
}
```

4.6 Hinweise und Fehlerbehandlung

Ergebnisse produzieren

- Der einmalige Aufruf einer `Check()`-Methode muss mindestens ein `Result` erzeugen.
- Ein Aufruf der `Check()`-Methode legt im Gut-Fall genau ein `Result` mit `Level.Info` an.
- Ein Aufruf der `Check()`-Methode legt im Schlecht-Fall mindestens ein `Result` mit `Level.Warning` oder `Level.Error` an, aber kein `Level.Info`.
- Kann ein Aufruf der `Check()`-Methode die Überprüfung nicht vollständig durchführen, zum Beispiel weil das Objekt nicht konsistent ist, ist dafür ein `Result` mit `Level.Warning` auszugeben.
- Für typische Ergebnismeldungen werden Standardtexte in der Klasse `DefaultMessage` bereitgestellt, wie zum Beispiel `DefaultMessage.Passed`.
- Geben Sie den Typ, Namen oder Pfad des Objekts nicht in `Result` eigenständig aus. Diese Informationen werden automatisch ergänzt; deshalb übergeben Sie beim Hinzufügen eines `Result` immer das geprüfte Objekt.
- Geben Sie Typ, Namen oder Pfad nur für Unterelemente des Objekts in `Result` mit aus, damit der Endanwender den Verstoß innerhalb des Objekts schneller finden kann.

Hinweis Da die Überprüfungen im Projekt Check in mehreren Threads parallel ausgeführt werden, ist davon auszugehen, dass zwei nacheinander hinzugefügte Ergebnisse nicht zwangsläufig direkt nacheinander in der Gesamtergebnisliste dargestellt werden.

Sandbox

Die Programmbibliotheken für die Regelsätze werden in einer Sandbox mit eingeschränkten Berechtigungen geladen. Dadurch wird das Sicherheitsrisiko durch das Ausführen von fremdem Code beim Endanwender reduziert.

Deshalb ist es auch nicht möglich, in einem Regelsatz auf das Openness-Objektmodell direkt zuzugreifen, sondern nur auf die bereitgestellten Abstraktionen.

Die folgenden Berechtigungen stehen einem Regelsatz zur Verfügung:

- SecurityPermission
  - SecurityPermissionFlag.Execution
- ReflectionPermission
  - ReflectionPermissionFlag.MemberAccess
- FileIOPermission
  - FileIOPermissionAccess.Read
  - FileIOPermissionAccess.PathDiscovery
  - Nur für das Verzeichnis des Projekt Check und dessen Unterverzeichnisse

Parallelisierung, Multithreading

Jede `Check()`-Methode muss eigenständig ausführbar sein. Instanzdaten sollten vermieden werden. Private (und im Idealfall statische) Methoden können genutzt werden, um Codeduplizierung zu vermeiden. Der Grund dafür ist, dass die Überprüfungen parallel in mehreren Threads ausgeführt werden und die Reihenfolge zufällig ist.

Fehlerbehandlung

Jede `Check()`-Methode wird automatisch mit einer Fehlerbehandlung umhüllt, daher müssen innerhalb der Methode keine allgemeinen Fehler via `try {…}` gefangen werden.

Des Weiteren stellt die Überprüfungsroutine sicher, dass beim Aufruf der `Check()`-Methode der Parameter `item` nicht `null` ist. Daher kann die Microsoft Design-Regel CA1062 \5\ ignoriert werden:

```
[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Design",
"CA1062:Validate arguments of public methods", MessageId = "0")]
```

Allerdings können manche Eigenschaften von `item` durchaus `null` sein, z. B. wenn die Daten nicht verfügbar sind. Deshalb müssen diese validiert werden.

JetBrains ReSharper

Die Klassen und Schnittstellen im Projekt Check SDK enthalten die Attribute `[CanBeNull]` und `[NotNull]` aus dem Namespace `JetBrains.Annotations`.

Damit erhalten Sie beim Entwickeln des Regelsatzes Hinweise auf:

- Eine übersehene `NullReferenceException`
  (ReSharper: "`Possible 'System.NullReferenceException'`")
- Unnötige Vergleiche gegen `null`
  (ReSharper: "`Expression is always true`")

5 Anhang

5.1 Service und Support

Industry Online Support

Sie haben Fragen oder brauchen Unterstützung?

Über den Industry Online Support greifen Sie rund um die Uhr auf das gesamte Service und Support Know-how sowie auf unsere Dienstleistungen zu.

Der Industry Online Support ist die zentrale Adresse für Informationen zu unseren Produkten, Lösungen und Services.

Produktinformationen, Handbücher, Downloads, FAQs und Anwendungsbeispiele – alle Informationen sind mit wenigen Mausklicks erreichbar:
support.industry.siemens.com

Technical Support

Der Technical Support von Siemens Industry unterstützt Sie schnell und kompetent bei allen technischen Anfragen mit einer Vielzahl maßgeschneiderter Angebote – von der Basisunterstützung bis hin zu individuellen Supportverträgen.

Anfragen an den Technical Support stellen Sie per Web-Formular:
www.siemens.de/industry/supportrequest

SITRAIN – Training for Industry

Mit unseren weltweit verfügbaren Trainings für unsere Produkte und Lösungen unterstützen wir Sie praxisnah, mit innovativen Lernmethoden und mit einem kundenspezifisch abgestimmten Konzept.

Mehr zu den angebotenen Trainings und Kursen sowie deren Standorte und Termine erfahren Sie unter:
www.siemens.de/sitrain

Serviceangebot

Unser Serviceangebot umfasst folgendes:

- Plant Data Services
- Ersatzteilservices
- Reparaturservices
- Vor-Ort und Instandhaltungsservices
- Retrofit- und Modernisierungsservices
- Serviceprogramme und Verträge

Ausführliche Informationen zu unserem Serviceangebot finden Sie im Servicekatalog:
support.industry.siemens.com/cs/sc

Industry Online Support App

Mit der App "Siemens Industry Online Support" erhalten Sie auch unterwegs die optimale Unterstützung. Die App ist für iOS und Android verfügbar:
support.industry.siemens.com/cs/ww/de/sc/2067

5.2 Links und Literatur

Tabelle 5-1 Themen

Nr.	Thema
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link auf die Beitragsseite des Anwendungsbeispiels https://support.industry.siemens.com/cs/ww/de/view/109741418
\3\	Programmierstyleguide für SIMATIC S7-1200 / S7-1500 https://support.industry.siemens.com/cs/ww/de/view/81318674
\4\	Microsoft Docs: Sicherheitstransparenter Code, Ebene 2 https://docs.microsoft.com/dotnet/framework/misc/security-transparent-code-level-2
\5\	Microsoft Design, CA1062: Argumente von öffentlichen Methoden validieren http://msdn.microsoft.com/library/ms182182.aspx
\6\	Microsoft Docs: Systemanforderungen für .NET Framework https://docs.microsoft.com/dotnet/framework/get-started/system-requirements
\7\	Microsoft: Download .NET Framework 4.8 https://dotnet.microsoft.com/download/dotnet-framework/net48
\8\	Lieferfreigabe TIA Portal Test Suite Advanced V16 https://support.industry.siemens.com/cs/ww/de/view/109775720

5.3 Änderungsdokumentation

Tabelle 5-2 Versionen

Version	Datum	Änderung
V1.0.0	10/2016	Erste Ausgabe, Name: "Programming style guide checker"
V1.1.0	11/2016	Version für TIA Portal V14
V1.1.1	04/2017	Version für TIA Portal V14 SP1
V1.5.0	04/2018	Version für TIA Portal V15
V1.5.1	09/2018	Version für TIA Portal V15.1
V2.0.0	n. v.	Erweiterungen
V3.0.0	08/2020	Vollständige Überarbeitung des Applikationsbeispiels • Neuer Name: "Projekt Check" • Ein Tool für mehrere TIA Portal-Versionen • Leichteres Erstellen eigener Regelsätze mithilfe des Projekt Check SDK (ohne Openness) • Ausleiten der Ergebnisse als Excel-Datei oder XML-Datei • Mehr TIA Portal-Objekte sind überprüfbar • Auswahl/Filtern der TIA Portal-Objekte und Regeln • Automatisierte Ausführung