VECTOR

CANdelaStudio24/25 Automation Pipeline on Windows

Version 1.0
2026-06-25
Application Note AN-IDG-1-018

Author Sauer, Ina

Restrictions Public Document

Abstract This document presents an approach for automating diagnostic development tasks using CANdelaStudio Server Edition (SE), thereby reducing the manual effort involved in updating diagnostic specifications such as CDD files. The automation pipeline includes validation through UDS/AUTOSAR checks as well as automated export of ODX and DEXT files, and it can be extended to support more advanced workflows involving testing and deployment. Implementation requires a source control system, a CI/CD platform, automation tools, and a server environment. Examples are provided for GitHub Actions and Jenkins.

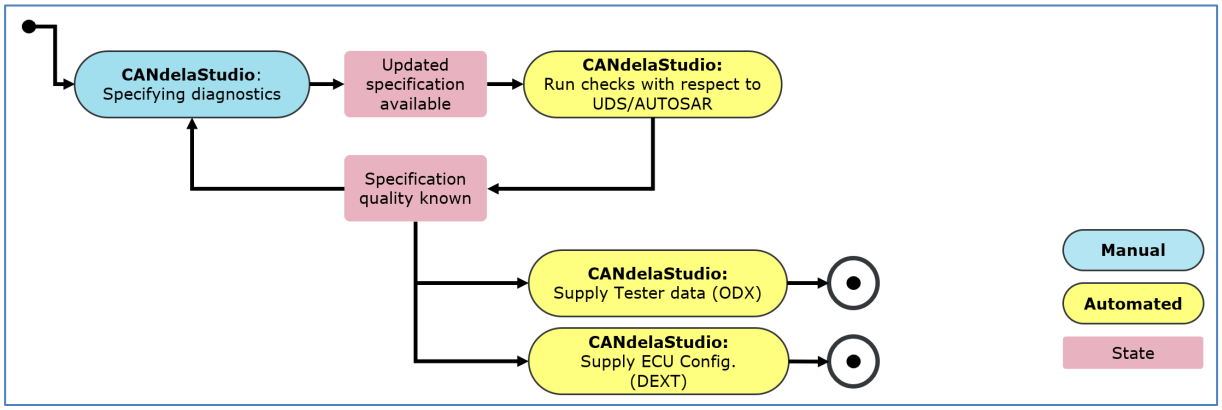
Table of Contents

1 Overview

During the diagnostic development process, updates to the diagnostic specification (CDD file) or to other data used in the toolchain are often required. This results in many manual steps.

With CANdelaStudio Server Edition (SE) and additional Vector tools, these manual steps can be automated.

The following picture shows an exemplary diagnostic automation process:



When the CDD file in the repository is updated, the automation process starts. In the first step, the UDS/AUTOSAR checker runs. If errors are detected, the data may need to be corrected manually.

The next two steps are exporting the updated ODX data and the DEXT file.

This is only a small and simple example. In practice, pipelines can be much more powerful and may include additional tools as well as testing and deployment steps.

Copyright © 2026 - Vector Informatik GmbH

Contact Information: www.vector.com or +49-711-80 670-0

2 Prerequisites

In general, the following components are required to create an automation pipeline:

- Source Control system for code/data (Git, Svn…)
- CI/CT/CD platform (system to control software building, testing, reporting and deployment), for example GitHub Actions, GitLab, Jenkins, …
- Tools for automation (CANdelaStudio SE, …)
- Server for executing the pipeline

This document describes how to use CANdelaStudio SE on a Windows server. As an example, it uses Git/GitHub together with GitHub Actions, with the source code and documents versioned in Git. Jenkins users can also find an example of pipeline automation described in a Jenkinsfile containing the corresponding pipeline steps.

Copyright © 2026 - Vector Informatik GmbH

Contact Information: www.vector.com or +49-711-80 670-0

3 CI-Server

We start with a VM (virtual machine) configured as a server with the following properties:

- OS: Windows Server 2022
- Core: 4 (Dual 2,7 GHz)
- Memory: 32GB
- Disk Size: C:\(OS) 300GB, D:\(Data) 100GB

PowerShell should already be installed.

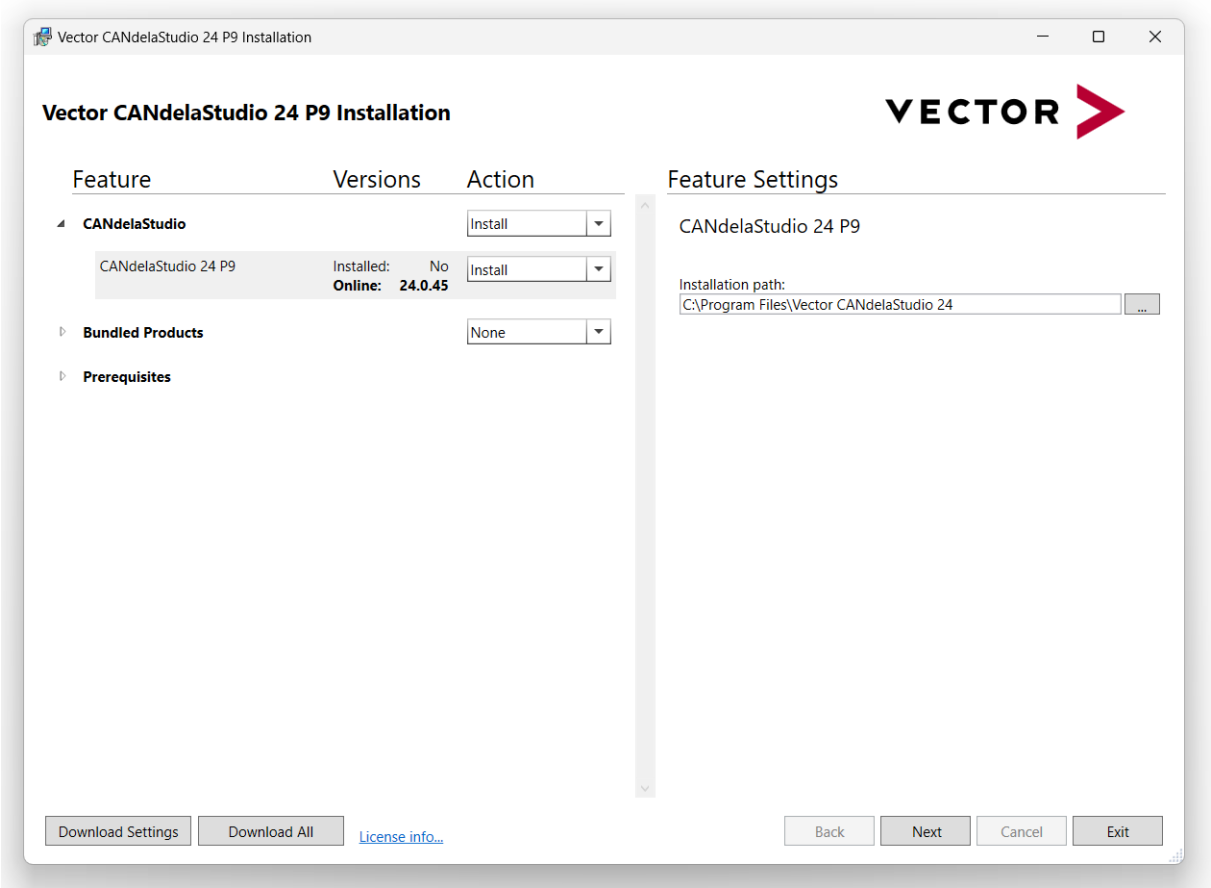
Copyright © 2026 - Vector Informatik GmbH

Contact Information: www.vector.com or +49-711-80 670-0

4 Installation on Server

- Install **CANdelaStudio** and activate the CANdelaStudio SE license with the help of the **Vector License Client.**

  To install CANdelaStudio SE, follow the instructions provided by the setup program (Setup.exe). The tool is installed in the default directory “C:\Program Files\Vector CANdelaStudio 24” (for version 24). The Vector License Client is installed automatically as well.



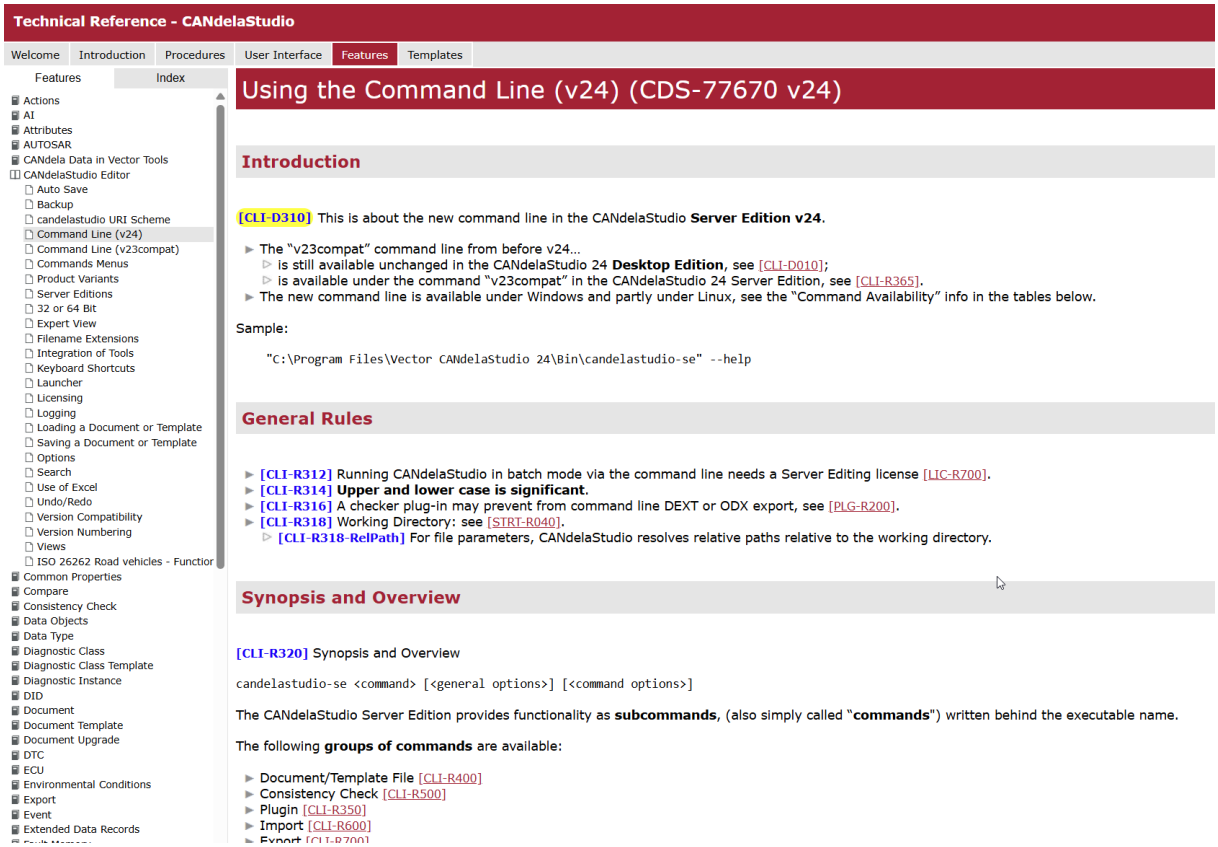
- Install **Git**

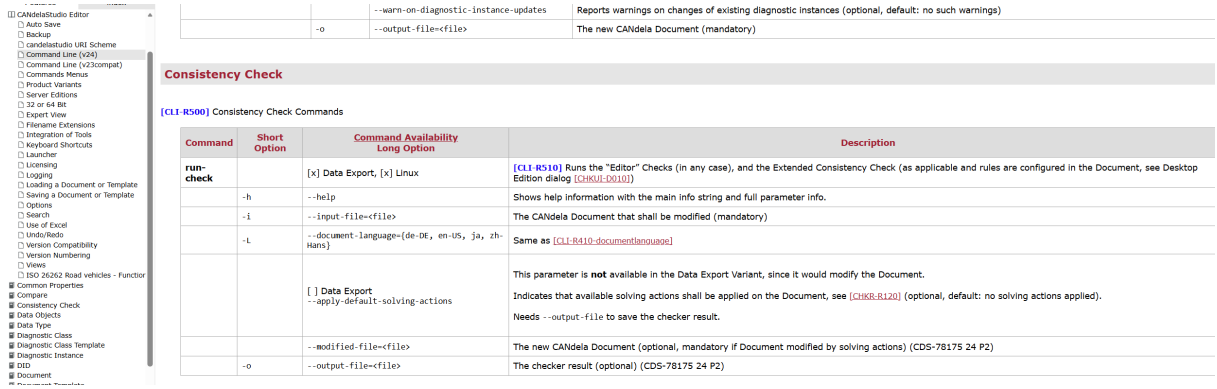
  Because the source code is retrieved from the GitHub repository to the server, Git must be installed on the server.

For more information, it is recommended to consult the CANdelaStudio Help (press F1 to open it from within CANdelaStudio):

Copyright © 2026 - Vector Informatik GmbH

Contact Information: www.vector.com or +49-711-80 670-0





Copyright © 2026 - Vector Informatik GmbH

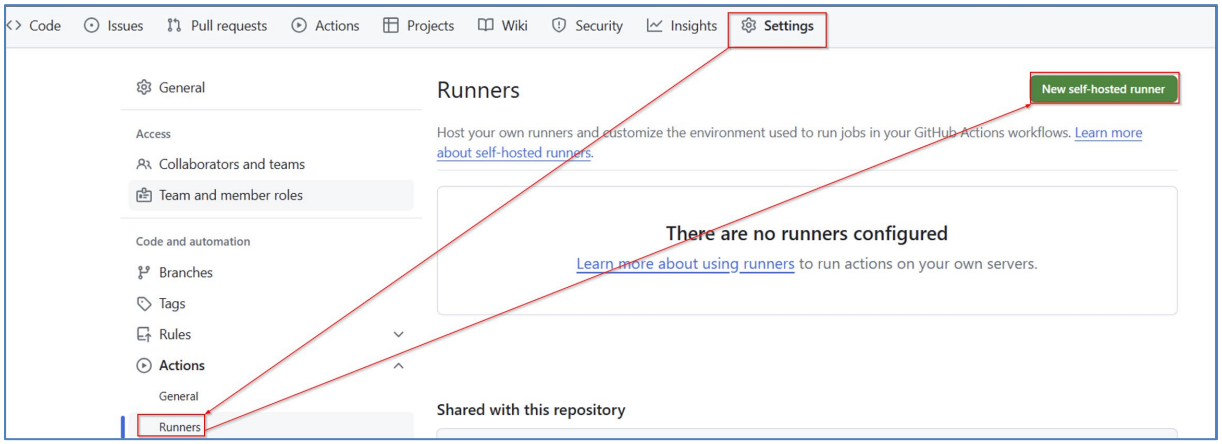
Contact Information: www.vector.com or +49-711-80 670-0

5 Pipelines

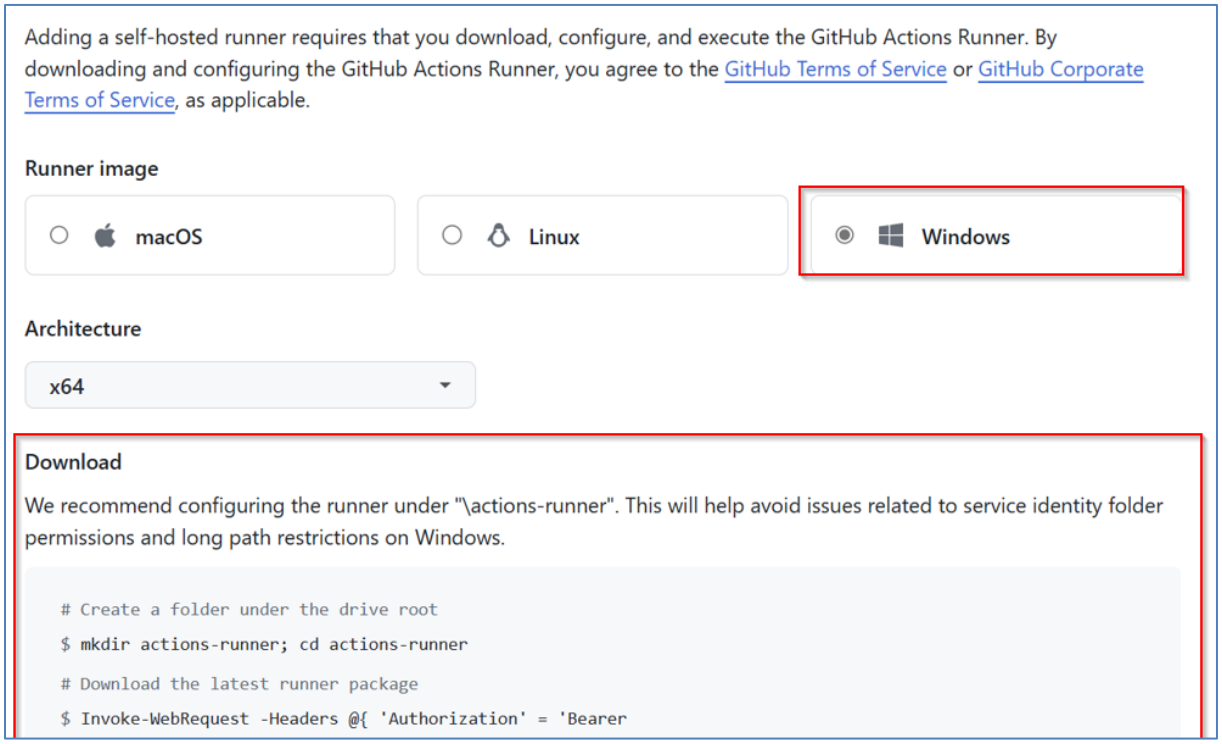
5.1 Pipeline with GitHub Actions

5.1.1 Adding a Runner

- Add “self-hosted runner” in your Github repository:
  Go to “Settings” -> “Actions” (under “Code and automation”) -> Runners -> “New self-hosted runner”



- Choose “Windows” and follow the steps to set up the runner (under **Download**, see picture below)



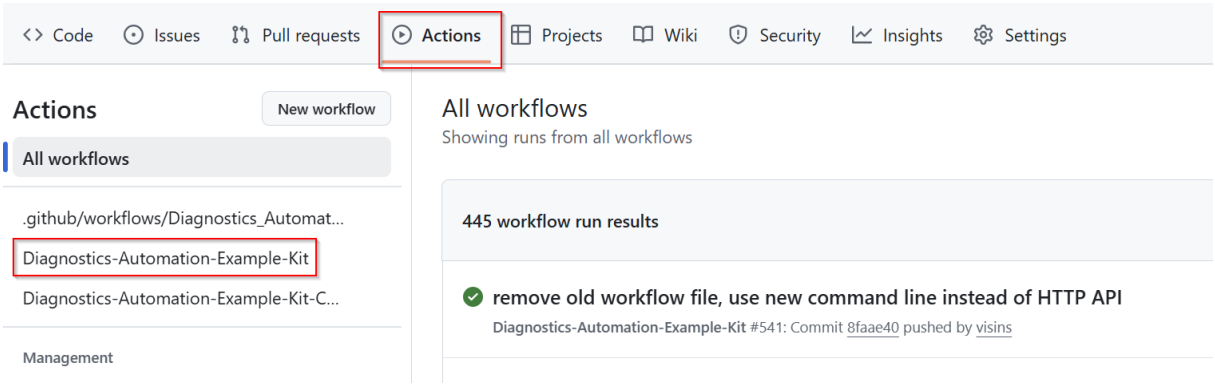
- The runner can be used in the “runs-on” command defined in the workflow file (see next chapter).

Copyright © 2026 - Vector Informatik GmbH

Contact Information: www.vector.com or +49-711-80 670-0

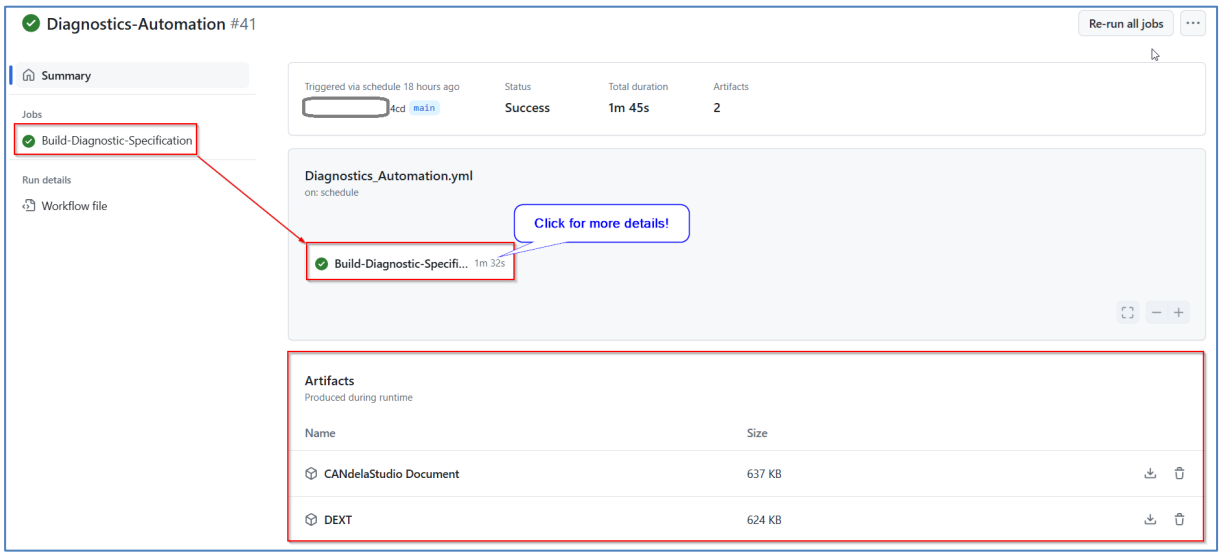
5.1.2 GitHub Actions GUI

In the GitHub Actions interface, all workflows are shown on the left side. In this example, we analyze the workflow named “Diagnostics-Automation-Example-Kit”.



The workflow runs are shown in the middle of the page. When you click a specific run, the log messages for that run and, if available, the generated artifacts are displayed. In the example shown, the CANdelaStudio document and the DEXT file are available for download.

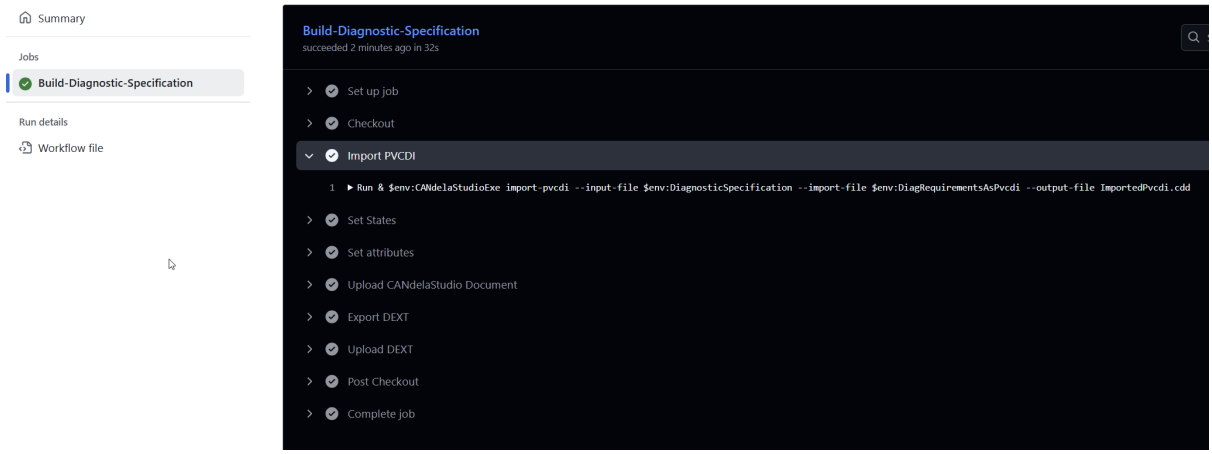
Overview of one workflow run.



Copyright © 2026 - Vector Informatik GmbH

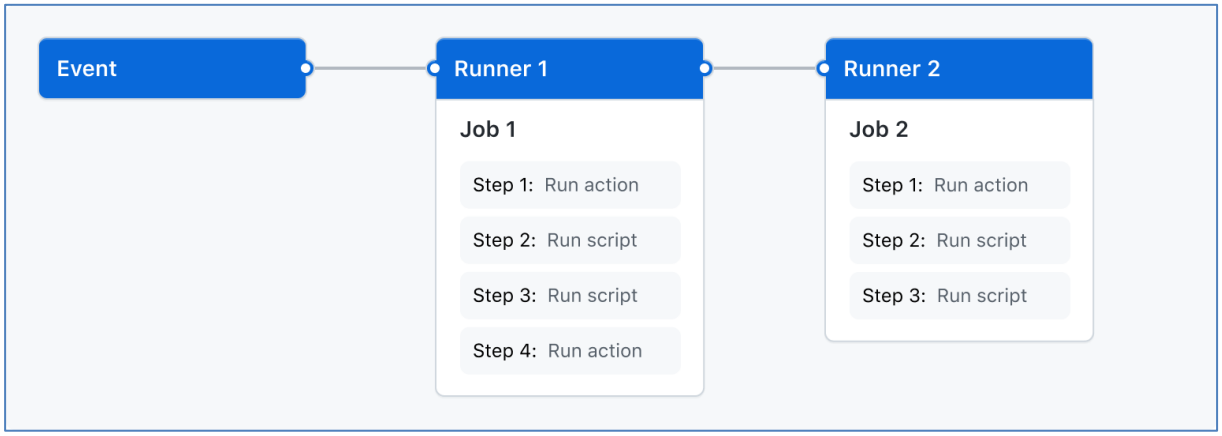
Contact Information: www.vector.com or +49-711-80 670-0

Details from the log:



5.1.3 Workflow Files

Workflow files are written in YAML (*.yaml, *.yml) and define the jobs and steps to be executed. When a specified event occurs, it triggers the configured jobs. These jobs may run on one or more runners. If multiple runners are used, jobs can be executed in parallel.



Reference: https://docs.github.com/en/actions/writing-workflows/about-workflows

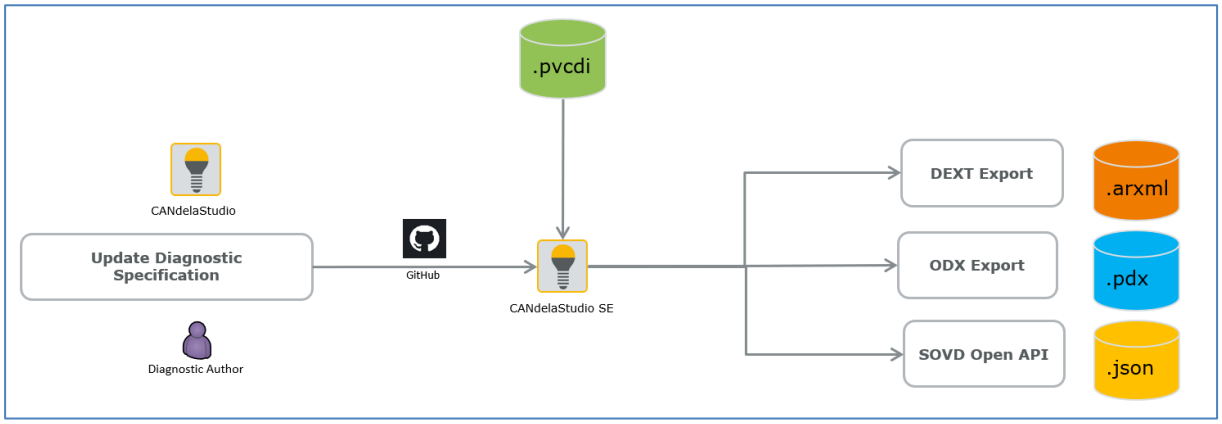
For more information see https://docs.github.com/en/actions

5.1.4 Example: Data Exchange

The following workflow uses important CANdelaStudio features such as PREEvision import (.pvcdi), DEXT export, ODX export, and SOVD export. The workflow is illustrated in the following figure.

Copyright © 2026 - Vector Informatik GmbH

Contact Information: www.vector.com or +49-711-80 670-0



The main CDD imports the .pvcdi file. Afterwards, three different export files are generated (.arxml, .pdx, and .json).

The repository contains a CDD file (ABS_ESP-Example.cdd), a .pvcdi file for import (Door.pvcdi), and the workflow file (Diagnostics_Automation_CLI.yml).

Content of the repository

When a change is pushed to GitHub, the workflow starts automatically. The workflow file is shown below:

```yaml
name: Diagnostics-Automation-Example-Kit-Cmd

on:
  push:

env:
  DiagnosticSpecification: 'ABS_ESP-Example.cdd'
  VariantQualifier: 'BaseVariant'
  DiagRequirementsAsPvcdi: '../Imports/Door.pvcdi'

  Dext: 'ABS_ESP-Example.arxml'
  DextExportLogFile: 'DextExportLogFile.log'

  PvcdiImportLogFile: 'PvcdiImportLogFile.log'

  Odx: 'ABS_ESP-Example.pdx'
  OdxExportLogFile: 'OdxExportLogFile.log'
```

Copyright © 2026 - Vector Informatik GmbH

Contact Information: www.vector.com or +49-711-80 670-0

```
  Sovd: 'ABS_ESP-Example.json'
  SovdExportLogFile: 'SovdExportLogFile.log'


jobs:
  Build-Diagnostic-Specification:
    runs-on: [ diagnostic-automation-test ]
    env:
      WORKING_DIR: Diag_Automation/Specification
      ServerInstallationDirectory: "C:/Program Files/Vector CANdelaStudio 24/Bin"
      CANdelaStudioExe: "C:/Program Files/Vector CANdelaStudio 24/Bin/candelastudio-se.exe"

    steps:
      - name: Checkout
        uses: actions/checkout@v4
        with:
          fetch-depth: 0

      - name: Import PVCDI
        working-directory: ${{ env.WORKING_DIR }}
        run: |
          & $env:CANdelaStudioExe import-pvcdi --input-file $env:DiagnosticSpecification --
import-file $env:DiagRequirementsAsPvcdi --output-file ImportedPvcdi.cdd

      - name: Upload CANdelaStudio Document
        uses: actions/upload-artifact@v3-node20
        with:
          name: CANdelaStudio Document
          path: ${{ env.WORKING_DIR }}/ABS_ESP-Example.cdd

      - name: Export DEXT
        working-directory: ${{ env.WORKING_DIR }}
        run: |
          & $env:CANdelaStudioExe export-dext --input-file $env:DiagnosticSpecification --
output-file $env:Dext --store-exported-dext-paths --modified-file
$env:DiagnosticSpecification --log-file $env:DextExportLogFile

      - name: Upload DEXT
        uses: actions/upload-artifact@v3-node20
        with:
          name: DEXT
          path: ${{ env.WORKING_DIR }}/ABS_ESP-Example.arxml

      - name: Export ODX
        working-directory: ${{ env.WORKING_DIR }}
        run: |
          & $env:CANdelaStudioExe export-odx --input-file $env:DiagnosticSpecification --
output-file $env:Odx --log-file $env:OdxExportLogFile
```

Copyright © 2026 - Vector Informatik GmbH

Contact Information: www.vector.com or +49-711-80 670-0

```
      - name: Upload ODX
        uses: actions/upload-artifact@v3-node20
        with:
          name: ODX
          path: ${{ env.WORKING_DIR }}/ABS_ESP-Example.pdx

      - name: Export SOVD
        working-directory: ${{ env.WORKING_DIR }}
        run: |
          & $env:CANdelaStudioExe export-sovd --input-file $env:DiagnosticSpecification --
variant-qualifier $env:VariantQualifier --output-file $env:Sovd --log-file
$env:SovdExportLogFile

      - name: Upload SOVD...
        uses: actions/upload-artifact@v3-node20
        with:
          name: SOVD
          path: ${{ env.WORKING_DIR }}/ABS_ESP-Example.json
```

The operations are executed through the new CANdelaStudio command line. We simply call `candelastudio-se.exe`, which is located in the default folder `C:/Program Files/Vector CANdelaStudio 24/Bin`. As an example, let us take a closer look at the step "**Import PVCDI**". The workflow calls `candelastudio-se.exe` (stored in the variable `CANdelaStudioExe`) with the operation to be performed, in this case `import-pvcdi`, and passes the input file stored in `DiagnosticSpecification`, the .pvcdi file to be imported stored in `DiagRequirementsAsPvcdi`, and the output file named "`ImportedPvcdi.cdd`".

The principle is the same for the other steps: the workflow file calls the CANdelaStudio command line with the required parameters.

5.2 Pipeline with Jenkins

The pipeline can also be run easily with Jenkins. The Jenkinsfile for this run can be derived easily from the GitHub Actions workflow file.

```
#!groovy

pipeline {
    agent { label 'diagnostic-automation-test' }

    environment {
        DiagnosticSpecification = 'ABS_ESP-Example.cdd'
        DiagRequirementsAsPvcdi = '../Imports/Door.pvcdi'

        Dext = 'ABS_ESP-Example.arxml'
        DextExportLogFile = 'DextExportLogFile.log'

        PvcdiImportLogFile = 'PvcdiImportLogFile.log'
```

Copyright © 2026 - Vector Informatik GmbH

Contact Information: www.vector.com or +49-711-80 670-0

```
        Odx = 'ABS_ESP-Example.pdx'
        OdxExportLogFile = 'OdxExportLogFile.log'

        Sovd = 'ABS_ESP-Example.json'
        SovdExportLogFile = 'SovdExportLogFile.log'

        WORKING_DIR = 'Diag_Automation/Specification'
        ServerInstallationDirectory = 'C:/Program Files/Vector CANdelaStudio 24/Bin'
        CANdelaStudioExe = 'C:/Program Files/Vector CANdelaStudio 24/Bin/candelastudio-
se.exe'
    }

    stages {

        stage('Import PVCDI') {
            steps {
                dir("${env.WORKING_DIR}") {
                    bat '"%CANdelaStudioExe%" import-pvcdi --input-file
"%DiagnosticSpecification%" --import-file "%DiagRequirementsAsPvcdi%" --output-file
ImportedPvcdi.cdd'
                }
            }
        }

        stage('Upload CANdelaStudio Document') {
            steps {
                archiveArtifacts artifacts: "${env.WORKING_DIR}/ABS_ESP-Example.cdd",
fingerprint: true
            }
        }

        stage('Export DEXT') {
            steps {
                dir("${env.WORKING_DIR}") {
                    bat '"%CANdelaStudioExe%" export-dext --input-file
"%DiagnosticSpecification%" --output-file "%Dext%" --store-exported-dext-paths --modified-
file "%DiagnosticSpecification%" --log-file "%DextExportLogFile%"'
                }
            }
        }

        stage('Upload DEXT') {
            steps {
                archiveArtifacts artifacts: "${env.WORKING_DIR}/ABS_ESP-Example.arxml",
fingerprint: true
            }
        }

        stage('Export ODX') {
```

Copyright © 2026 - Vector Informatik GmbH

Contact Information: www.vector.com or +49-711-80 670-0

```
            steps {
                dir("${env.WORKING_DIR}") {
                    bat '"%CANdelaStudioExe%" export-odx --input-file
"%DiagnosticSpecification%" --output-file "%Odx%" --log-file "%OdxExportLogFile%"'
                }
            }
        }

        stage('Upload ODX') {
            steps {
                archiveArtifacts artifacts: "${env.WORKING_DIR}/ABS_ESP-Example.pdx",
fingerprint: true
            }
        }

        stage('Export SOVD') {
            steps {
                dir("${env.WORKING_DIR}") {
                    bat '"%CANdelaStudioExe%" export-sovd --input-file
"%DiagnosticSpecification%" --variant-qualifier "%VariantQualifier%" --output-file "%Sovd%"
--log-file "%SovdExportLogFile%"'
                }
            }
        }

        stage('Upload SOVD') {
            steps {
                archiveArtifacts artifacts: "${env.WORKING_DIR}/ABS_ESP-Example.json",
fingerprint: true
            }
        }
    }
}
```

Copyright © 2026 - Vector Informatik GmbH

Contact Information: www.vector.com or +49-711-80 670-0

6 Tips

- If the server has been updated or the configuration has changed, restart the runner.
- If there are problems connecting to the GitHub server (for example, the error “self signed certificate in certificate chain”), set the Windows environment variable “NODE_EXTRA_CA_CERTS”. It should contain the path to the certificate file (.crt).

7 Contacts

For a full list with all Vector locations and addresses worldwide, please visit https://vector.com/contact/.

Copyright © 2026 - Vector Informatik GmbH

Contact Information: www.vector.com or +49-711-80 670-0