



User Guide

ARCAD CodeChecker

Version 24.0



Publication Date: January, 2024

Prepared by the ARCAD Documentation Team

North America & LATAM

1 N. State St, 15th Floor
Chicago, IL
USA
1-603-371-9074
1-603-371-3256 (support calls only)
sales-us@arcadsoftware.com

EMEA (HQ)

55 Rue Adrastée – Parc Altaïs
74650 Chavanod/Annecy
France
+33 450 578 396
sales-eu@arcadsoftware.com

Asia Pacific

5 Shenton Way #22-04
UIC Building
Singapore 068808
sales-asia@arcadsoftware.com

Copyright © 1992-2024 by ARCAD. All rights reserved.

The following terms are names owned by International Business Machines Corporation in the United States, other countries, or both: AS/400®, ClearCase, ClearQuest®, DB2, DB2 Connect™, DB2 Universal Database™, ibm.com, IBM i, iSeries, System i, OS/400, Rational®, SP2, Service Pack, WebSphere. Java and all names based on Java are owned by Oracle Corp. in the United States, other countries, or both. Eclipse is a registered trademark of Eclipse Foundation, Inc. Other names of companies, products or services are the property of their respective owners.

Contact ARCAD

Headquartered in France at the foot of the Alps, ARCAD offers global services and has offices and partners all over the world. ARCAD partners with leading-edge companies throughout the world to offer full services, close to home.

Visit our website to [Contact Us](#) and find out more about our company and partners, or to request a demo.

The [ARCAD Customer Portal](#) is intended for current and potential customers that have full or trial versions of ARCAD software. If you already use or are interested in using an ARCAD product, the portal lets you view all of your current licenses and generate your own temporary license keys for most ARCAD products. It grants you access to the ARCAD product knowledge base (new releases, release notes and current documentation).

Do you have a request for change or have you encountered a bug? Log into the [ARCAD Helpdesk](#) and create a ticket.

ARCAD guarantees consultant support 24 hours a day, 5 days a week (24/5) to registered members. Calls received are redirected, according to the hour, to put you in contact with a support team in or near your timezone.

Country	Address	Account Contact	Support Contact
France	ARCAD Software (HQ) 55 Rue Adrastée 74650 Chavanod ARCAD Software 17 chemin de la plaine 07200, Saint-Didier-sous-Aubenas		Worldwide 24/7: +1 603 371 3256 France only: +33 450 57 28 00 support@arcadsoftware.com ARCAD Helpdesk
Germany	ARCAD Software Deutschland GmbH c/o Pramex International GmbH Savignystr. 43, 60325 Frankfurt am Main		
China	ARCAD Software #2035, Yuehai Plaza, 180 Wanbo 2nd Road, Nancun, Panyu District, Canton	+86 (020)22324643 +86 (020)22324649 sales-asia@arcadsoftware.com	
India	ARCAD Software D-280/281/282, Vibhuti Khand Gomti Nagar, Lucknow		
Singapore	ARCAD Software 5 Shenton Way #22-04 UIC Building Singapore 068808		
USA	ARCAD Software 1 N. State St, 15th Floor Chicago, IL	+1 (603) 371-9074 +1 (603)-371-3256 (support calls only) sales-us@arcadsoftware.com	

Table 1: Contact ARCAD

Preface

Document purpose

This document is intended to guide users through configuring and using the following ARCAD CodeChecker module(s):

- The CodeChecker Server
- The CodeChecker Studio
- The ARCAD CodeChecker plug-in for RDi
- The ARCAD CodeChecker plug-in for Jenkins
- The ARCAD CodeChecker plug-in for SonarQube

Intended audience

This document is intended for administrators, development team managers, developers, or any person in charge of ensuring or monitoring an application's code quality.

Publication record

Unless stated otherwise, all content is valid for the most current version of ARCAD CodeChecker listed as well as every subsequent version.

Product version	Document version	Publication Date	Update record
≥ 24.0	4.0	January, 2024	Documentation updated for product evolution New features and enhancements: <ul style="list-style-type: none"> • Added new options for rules and rule sets: duplicate rule sets, copy paste rules, rules categories, etc. • Added new external target to run code quality check with Jenkins and the CLI on code managed externally.
23.3	3.9	October, 2023	Documentation updated for product evolution New features and enhancements: <ul style="list-style-type: none"> • Added new Reference Failure Level in Quality Rules to report reference issues • Added new Batch Update option to apply modifications to multiple rules at once • Added new options in RDi preferences to disable delta comparisons between components during executions on ARCAD Skipper
23.2	3.8	July, 2023	Documentation updated for product evolution. New features and enhancements: <ul style="list-style-type: none"> • Added Git targets • Added option to generate a project folder used to invoke the Sonar Scanner externally and upload Campaigns results, using the CLI or the Jenkins plug-in. • Reorganization of user roles

Table 2: ARCAD CodeChecker User Guide publication record

Product version	Document version	Publication Date	Update record
23.1	3.7	April, 2023	No functional changes Minor documentation corrections
23.0	3.6	January, 2023	Documentation updated for product evolution. New features and enhancements: <ul style="list-style-type: none"> • Added CLI option to write the Target issues in a file after a campaign execution • Added option to reset a Target's issues before executing a Campaign with the CLI and the Jenkins plug-in • Added option to display the execution status and errors for a campaign in the CodeChecker Studio • Added option to end a CLI or Jenkins plug-in execution if an error occurs

Table 2: ARCAD CodeChecker User Guide publication record

Related documentation





ARCAD technical documentation can be accessed from the product's online help or by logging into the [Customer Portal](#) on our website.

ARCAD Glossary
ARCAD CodeChecker Installation Guide

Table 3: Related documentation

About the entity access tables

At the beginning of most of the chapters in this document, an entity access table displays important information regarding the entity described in the chapter.

Required role	 Rule Management
Access	 CodeChecker Server →  Rules Configuration →  Quality Rules

1. The role(s) required for the user's profile in order to access, create, edit, delete, manage, etc. anything concerning the entity described.

Reference

For more information about user roles, refer to [User roles on page 44](#) in the appendices.

2. The path to access the entity in the CodeChecker Studio.

Reference

For more information about how to navigate inside the CodeChecker Studio, refer to [Overview on page 28](#).

Contents

Contact ARCAD	3
Preface	4
Contents	6
Tables	11
Figures	12

ABOUT ARCAD CODECHECKER

1 Presentation of ARCAD CodeChecker	15
1.1 About the business context.....	16
1.2 About the CodeChecker Server.....	16
1.3 About the CodeChecker Studio.....	17
1.4 About the ARCAD CodeChecker plug-ins.....	17
2 Main concepts	19

GETTING STARTED WITH THE CODECHECKER STUDIO

Introduction to the CodeChecker Studio	21
3 Managing the CodeChecker Server	22
3.1 Configuring the CodeChecker Server connection.....	23
3.1.1 Configuring the server connection properties.....	23
3.2 Connecting to the CodeChecker Server.....	24
3.3 Configuring the TLS settings.....	25
3.4 Loading default rule sets.....	26
3.5 Changing the password of a user account.....	26
3.6 Verifying CodeChecker Server versions.....	27
3.7 Viewing CodeChecker Studio logs.....	27
4 Overview	28

CONFIGURING ARCAD CODECHECKER

Introduction to ARCAD CodeChecker configuration	31
5 Registering license keys	32
6 Preferences	33
6.1 General preferences.....	33
6.2 Mail server preferences.....	34
6.3 Execution preferences.....	35
6.4 ARCAD Skipper precompiler preferences.....	35
6.5 Sonar Integration preferences.....	36

USER ADMINISTRATION

Introduction to user administration	39
--	-----------

7 Users	40
7.1 Creating users	40
7.2 Importing users	41
7.3 Editing users	42
7.4 Deleting users	43
8 User roles	44

CONNECTIONS

Introduction to connections	46
9 IBM i servers	47
9.1 Creating IBM i server connections	47
9.2 Set a default IBM i server	48
9.3 Deleting IBM i server connections	49
10 SSH Keys	50
10.1 Creating SSH keys	50
10.1.1 Create new SSH keys	50
10.1.2 Import existing SSH keys	51
10.2 Editing SSH keys	51
10.3 Accessing the public key	51
10.4 Deleting SSH keys	52

QUALITY RULES CONFIGURATION

Introduction to quality rules	54
11 Rule sets	55
11.1 Creating rule sets	56
11.2 Editing rule sets	56
11.3 Activating rule sets	58
11.4 Loading default rule sets	59
11.5 Open the rule set documentation	59
11.6 Exporting and importing rule sets	60
11.6.1 Exporting rule sets	60
11.6.2 Importing rule sets	60
11.7 Deleting rule sets	61
12 Quality rules	62
12.1 Creating quality rules	62
12.2 Editing quality rules	63
12.3 Quality rules batch update	66
12.4 Documenting quality rules	66
12.5 Activating quality rules	67
12.6 Adding quality rules to a rule set	68
12.7 Deleting quality rules	68
13 Metrics	69

13.1 Creating metrics	70
13.2 Editing metrics	70
13.3 Deleting metrics	72
14 Metric models	73
14.1 Creating metric models	74
14.2 Editing metric models	74
14.3 Deleting metric models	77
15 Validation expressions	78
15.1 Creating validation expressions	79
15.2 Editing validation expressions	79
15.3 Deleting validation expressions	81

CAMPAIGNS CONFIGURATION

Introduction to code analysis campaigns	83
16 Type aliases	84
16.1 Creating type aliases	85
16.2 Editing type aliases	85
16.3 Deleting type aliases	85
17 Code reviews	86
17.1 Creating code reviews	86
17.2 Editing code reviews	87
17.2.1 Code Review	88
17.2.2 Measures	89
17.3 Deleting code reviews	90
18 Targets	91
18.1 Creating targets	92
18.2 Editing targets	92
18.2.1 IBM i targets	94
18.2.2 ARCAD targets	97
18.2.3 Git targets	99
18.2.4 External targets	101
18.3 Resetting targets	102
18.4 Deleting targets	102

EXECUTING CAMPAIGNS

Introduction to code quality campaigns executions	104
19 Campaigns	105
19.1 Creating campaigns manually	106
19.2 Duplicating campaigns	107
19.3 Launching campaigns manually	107
19.4 Understanding campaign results	108
19.5 Exporting campaign results	112
19.6 Emailing campaign results	113

19.7 Deleting campaigns	113
20 Schedules	114
20.1 Creating schedules	114
20.2 Editing schedules	115
20.3 Starting and stopping schedules	119
20.4 Understanding scheduled campaign results	119
20.5 Deleting schedules	119
21 Issues	120
22 The Command Line Interface	122

ARCAD CODECHECKER PLUG-INS

Introduction to the ARCAD CodeChecker plug-ins	128
23 ARCAD CodeChecker for RDi	129
23.1 Connecting to the CodeChecker Server	129
23.1.1 Editing TLS Settings	131
23.2 Accessing the RDi plug-in	131
23.3 Setting rules execution preferences	132
23.4 Displaying rule sets	133
23.5 Executing quality rules	134
23.5.1 Executing all the quality rules	135
23.5.2 Selecting and execute quality rules	135
23.5.3 Executing the last-used quality rules sets	135
23.6 Managing executions	136
23.6.1 Exporting issues	136
23.6.2 Re-executing past issues	137
23.7 Managing error lists	137
23.7.1 Understanding an error list	137
23.7.2 Correcting failures from an error list	138
24 ARCAD CodeChecker for Jenkins	139
24.1 Prerequisites	139
24.2 Configuring the ARCAD CodeChecker for Jenkins plug-in	139
24.2.1 Adding the plug-in to Jenkins	139
24.2.2 Setting the connection details	140
24.3 Setting up an ARCAD CodeChecker project	140
24.3.1 Freestyle projects	140
24.3.2 Pipeline projects	142
24.4 Executing campaigns in Jenkins	145
25 ARCAD CodeChecker for SonarQube	146
25.1 Enabling results upload for a target	146
25.2 Accessing the results in SonarQube	147

APPENDICES

Dashboard metrics	149
--------------------------------	------------

Glossary	151
-----------------------	------------

Tables

Table 1: Contact ARCAD.....	3
Table 2: ARCAD CodeChecker User Guide publication record.....	4
Table 3: Related documentation.....	5
Table 4: The user roles.....	44
Table 5: The TLS Settings.....	131

Figures

Figure 1: ARCAD CodeChecker in the ARCAD product suite	16
Figure 2: The CodeChecker Studio for RDi	22
Figure 3: Configure the server connection	23
Figure 4: The server Connection Properties dialog	23
Figure 5: The CodeChecker Studio connection dialog	24
Figure 6: Reload default Rule Sets	26
Figure 7: The CodeChecker Studio	28
Figure 8: The ARCAD CodeChecker navigator	29
Figure 9: Register the ARCAD CodeChecker license key	32
Figure 10: Mail server settings	34
Figure 11: Execution preferences	35
Figure 12: ARCAD Skipper precompiler settings	35
Figure 13: The SonarQube settings	36
Figure 14: The Administration node in the navigator	39
Figure 15: User Management	40
Figure 16: IBM i Servers	47
Figure 17: The IBM i server editor	48
Figure 18: The Rules Configuration node in the navigator	54
Figure 19: Rule Sets	55
Figure 20: The Rule Set editor	57
Figure 21: Reload default Rule Sets	59
Figure 22: The rule sets documentation	59
Figure 23: The Rule Set import dialog	61
Figure 24: Quality Rules	62
Figure 25: The Quality Rule editor	63
Figure 26: The Quality Rule Documentation editor	67
Figure 27: Metrics	70
Figure 28: The Metric editor	71
Figure 29: Metric Models	73
Figure 30: The Metric Model editor	75
Figure 31: The Metric Model's Script editor	76
Figure 32: Validation Expressions	78
Figure 33: The Validation Expression editor	80
Figure 34: The Campaign Configuration node in the navigator	83
Figure 35: Type aliases	84
Figure 36: The Type Alias edition dialog	85
Figure 37: Code Reviews	86
Figure 38: The Code Review editor	88
Figure 39: Targets	91
Figure 40: The Target editor - IBM i targets	94
Figure 41: The Target editor - ARCAD targets	97
Figure 42: The Target editor - Git targets	99
Figure 43: The Target editor - External targets	101
Figure 44: The Campaign Execution node in the navigator	104
Figure 45: Campaigns	106
Figure 46: The Campaign Results view	108
Figure 47: The Campaign Results execution details	110
Figure 48: The source member's code view	111

Figure 49: The Campaign Results log tab.....	112
Figure 50: The campaign Excel export file.....	112
Figure 51: Schedules.....	114
Figure 52: The Schedule editor.....	116
Figure 53: Issues.....	120
Figure 54: Review issues.....	121
Figure 55: The CodeChecker Server connection preferences in RDi.....	130
Figure 56: The ARCAD CodeChecker plug-in for RDi.....	132
Figure 57: The Preference dialog in RDi - Rules Execution.....	132
Figure 58: The Rule Sets view in RDi.....	133
Figure 59: The Rules execution view in RDi.....	136
Figure 60: The Execution Logs view in RDi.....	136
Figure 61: The Error List view in RDi.....	137
Figure 62: The ARCAD CodeChecker code review build step in Jenkins.....	141
Figure 63: The campaign results in Jenkins.....	145



ABOUT ARCAD CODECHECKER

1 Presentation of ARCAD CodeChecker

ARCAD CodeChecker automates the tasks of analyzing code quality, detecting complexity hotspots in your code, and ensuring a consistent level of quality throughout your application.

To determine code quality, ARCAD CodeChecker uses quality rules, metrics and metric models. A metric is a numeric value deduced from the analysis of a source code. A metric is based on a metric model, which defines the actions to carry out to retrieve a value during the code review process. Metrics are then used in conditional expressions called quality rules. If the metrics respect the conditions set by the quality rules, then the rules succeed. However, if the metrics do not respect the conditions, the quality rules fail and the source code analyzed does not meet the expected quality standards.

To carry out the code review process, you must associate a code review with a target to create a campaign. A code review enables you to define the quality rules to execute, and a target specifies the application and the source members on which these rules must be executed. When creating a campaign, a code review is associated with a target to define the context of the code analysis. When you launch a campaign, each source member contained in the defined target application is analyzed to determine if it complies with the quality rules specified, thus indicating the code quality of your applications.

ARCAD CodeChecker lets you easily define customized quality rules and metrics to measure code quality, and enables you to carry out the code review process during several development steps (coding, continuous integration, quality processes).

ARCAD CodeChecker analyzes applications written in the main languages used in IBM i environments:

- CL
- COBOL
- RPG (RPG III)
- RPGLE (RPG IV)

ARCAD CodeChecker is designed as a suite of tools:

- The CodeChecker Server stores all the entities required to set up and execute a code review process, and gives you access to all the standard predefined entities.
- The CodeChecker Studio enables you to access, create and manage all the entities required to set up and execute a code review process.
- The ARCAD CodeChecker plug-ins are used to execute quality rules directly on the source code, or to analyze the results of the code review / target associations.

ARCAD CodeChecker is also available as a service provider for external applications.

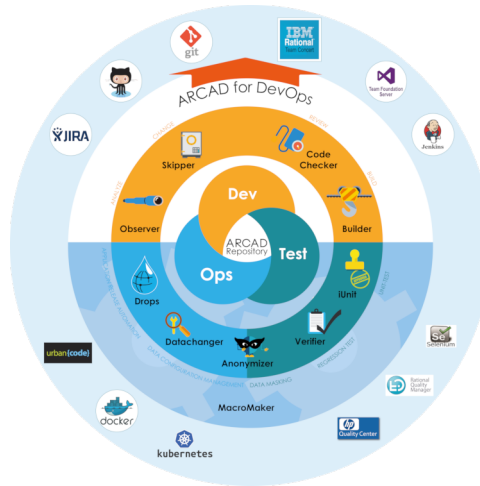


Figure 1: ARCAD CodeChecker in the ARCAD product suite

1.1 About the business context

Development cycles are getting shorter and shorter, and agile software development is used by more and more companies around the world. Development teams must be reactive and able to deliver functional versions at any time. Therefore, the source code of an application must be easily maintainable in order to quickly identify and correct critical errors. The whole idea is to be able to put the product on the market as early as possible while maintaining code quality.

Code quality is like an insurance policy for your application. By limiting source code complexity, you will automatically:

- reduce the testing workload,
- reduce the time needed to understand code,
- safeguard the reliability and robustness of your application.

Measure and study the quality of your application's source code by generating reports that provide detailed accounts of how it does or does not comply with the quality rules defined. High quality source code will extend the lifespan of your application and thus leverage your investment.

1.2 About the CodeChecker Server

The CodeChecker Server stores all the elements required to set up and execute a code review process, and gives access to the standard metric models and other predefined entities that are delivered with ARCAD CodeChecker.

The CodeChecker Server is also in charge of executing the code review process. When a campaign is launched, the server executes the active quality rules on the defined target application. The server analyzes the code quality and generates results, which can be viewed in the CodeChecker Studio or in the ARCAD CodeChecker plug-ins.

1.3 About the CodeChecker Studio

The CodeChecker Studio is an Eclipse-based application that connects to the CodeChecker Server. The studio is available as:

1. A stand-alone Rich Client Platform (RCP),
2. An RDi perspective.

Note

Both versions of the studio have the same interface, offer the same features and use the CodeChecker Server the same way. However, the RDi perspective must be installed as a plug-in on RDi, whereas the RCP is a stand-alone application.

The CodeChecker Studio enables you to:

- create and manage all the entities required to set up a code review process (rule sets, quality rules, metrics, metric models and validation expressions),
- create and manage all the entities required to execute a code review process (code reviews, targets, campaigns and schedules),
- activate specific rule sets and quality rules so they can be used in the studio or in the plug-ins.

Reference

For more information about the studio refer to [Managing the CodeChecker Server on page 22](#).

1.4 About the ARCAD CodeChecker plug-ins

ARCAD CodeChecker is available as a plug-in for integrated modules.

The ARCAD CodeChecker plug-in for RDi

The ARCAD CodeChecker plug-in for RDi is designed for the RDi integrated development environment (IDE).

The plug-in is designed for developers working in the RDi IDE. It enables developers to test the quality of the source code they are working on by executing active rule sets or specific quality rules, and know instantly if the source code does not comply with the standards set for the development team. With the plug-in, developers can easily know which quality rules are failing and identify which part of their source code is causing the failure. The portions of code causing the failures are clearly identified, making it easy for developers to modify their source code to comply with the quality standards.

The ARCAD CodeChecker plug-in for Jenkins

The ARCAD CodeChecker for Jenkins plug-in is intended for development team managers, or any person in charge of the evolution of an application's source code. It enables you to test the quality of a build and generate an analysis report for the build. With this report, you can quickly identify the files that do not comply with the quality rules executed during the code review process. The plug-in also enables you to stop the build process if a FATAL quality rule is failing.

The ARCAD CodeChecker plug-in for SonarQube

The ARCAD CodeChecker solution for SonarQube integrates the results of code quality campaign in SonarQube, the open-source platform for continuous inspection of code quality. You can review issues raised in a campaign with the functionalities offered by the Sonar interface (code review, statistics, etc.). Like in the CodeChecker Studio, rule documentation is available for each issue in Sonar.

The solution is composed of:

- The ARCAD CodeChecker plug-in for SonarQube, to be installed on the SonarQube server. The plug-in is a SonarQube add-on that allows you to manage your source code quality with SonarQube.
- The Sonar Scanner, supplied by Sonar and fully managed on the CodeChecker Server side. The scanner is a Sonar component that ensures the communication of the CodeChecker Server with SonarQube.

2 Main concepts

ARCAD CodeChecker is intended to help you monitor and improve the code quality of your applications.

The main steps required to set up and carry out the code review process are described below. It is not required to follow the order in which the steps are described.

1. Start the CodeChecker Studio and connect to the CodeChecker Server to begin working with ARCAD CodeChecker. If needed, configure the studio and the server.

[Managing the CodeChecker Server on page 22](#) and [Preferences on page 33](#)

2. Create user accounts and assign the appropriate roles for each person in your team who will be involved during the code review process and who will need to use the CodeChecker Studio.

[Users on page 40](#)

3. Set up the code review process by creating and defining all the relevant entities: the connection to the machine where your code is, the rule sets of quality rules, code reviews and targets, etc. All these entities enable you to define the code quality standards to reach on your code.

[Introduction to connections on page 46](#), [Introduction to quality rules on page 54](#) and [Introduction to code analysis campaigns on page 83](#)

4. Execute the code review process by associating a code review with a target to manually create and launch a campaign. Campaigns can also be created automatically using schedules. When a campaign is finished, analyze the results to check if the code quality standards to reach are achieved, and to determine what needs to be improved.

[Introduction to code quality campaigns executions on page 104](#)

5. Alternatively, use the ARCAD CodeChecker plug-ins for additional features:

- Use the ARCAD CodeChecker plug-in for RDi to directly check the quality of the source code you are working on and know instantly if your code complies with the code quality standards.

[ARCAD CodeChecker for RDi on page 129](#)

- Use the ARCAD CodeChecker plug-in for Jenkins to test the quality of a build and generate an analysis report for the build. This report enables you to quickly identify the source code that does not comply with the code quality standards.

[ARCAD CodeChecker for Jenkins on page 139](#)

- Use the ARCAD CodeChecker plug-in for SonarQube to see the evolution of the quality of an application's source code throughout the development process.

[ARCAD CodeChecker for SonarQube on page 146](#)



GETTING STARTED WITH THE CODECHECKER STUDIO

Introduction to the CodeChecker Studio

This section is intended to help you get started with using the CodeChecker Studio and make sure ARCAD CodeChecker is correctly configured before setting up the code review process.

This section describes how to start the CodeChecker Studio and configure its CodeChecker Server connection.

3 Managing the CodeChecker Server

Chapter Summary

- 3.1 Configuring the CodeChecker Server connection..... 23
- 3.2 Connecting to the CodeChecker Server..... 24
- 3.3 Configuring the TLS settings..... 25
- 3.4 Loading default rule sets..... 26
- 3.5 Changing the password of a user account..... 26
- 3.6 Verifying CodeChecker Server versions..... 27
- 3.7 Viewing CodeChecker Studio logs..... 27

The CodeChecker Studio must connect to the CodeChecker Server to access and store all the elements required for carrying out a code review process.


The CodeChecker Studio can either be accessed:


- as a stand-alone RCP,
- as a perspective in RDi.

The features are identical no matter how the products is accessed.

 **Reference**

For more information about installing the CodeChecker Studio and server, refer to the ARCAD CodeChecker Installation Guide.

To launch the stand-alone RCP, either double-click on the desktop icon or select  **CodeChecker Studio** from the Start menu or the toolbar.

To access the perspective in RDi, open the **Window** menu then select **Open Perspective > Other...** Select  **CodeChecker Studio**, then click **OK**.

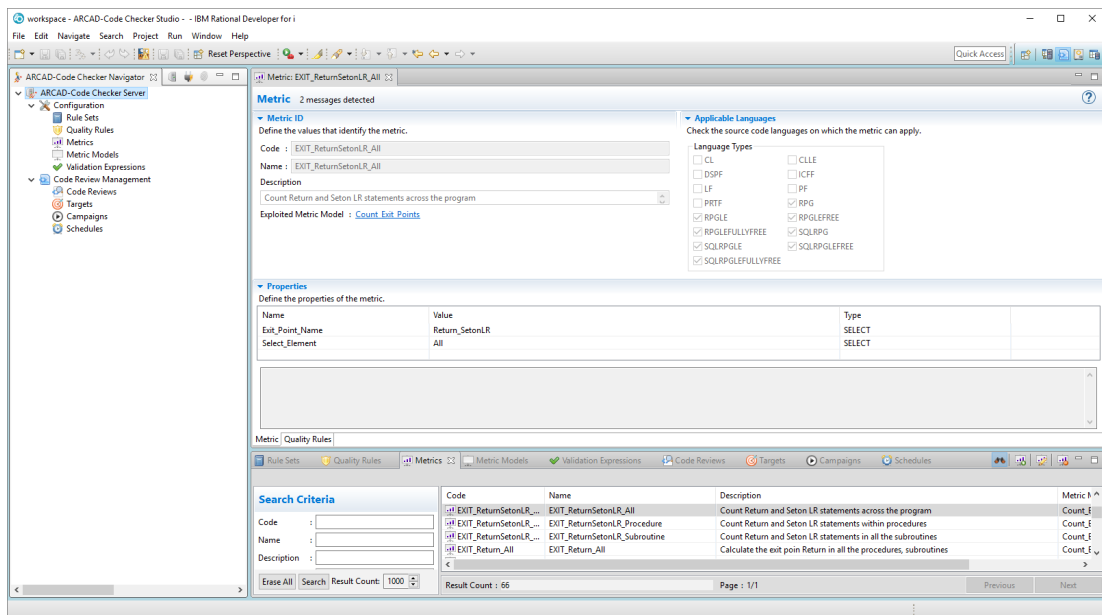



Figure 2: The CodeChecker Studio for RDi

3.1 Configuring the CodeChecker Server connection

The CodeChecker Studio must connect to the CodeChecker Server to access and store all the elements required for carrying out a code review process. Configuration options for the CodeChecker Server are managed in the **Connection properties** window.

After installing a new server or studio and the product is launched for the first time, the **Connection Properties** wizard opens automatically.

If the CodeChecker Studio should use new connection information to connect to the CodeChecker Server, click the  CodeChecker Server icon in the navigator to open the **Connection Properties** dialog.

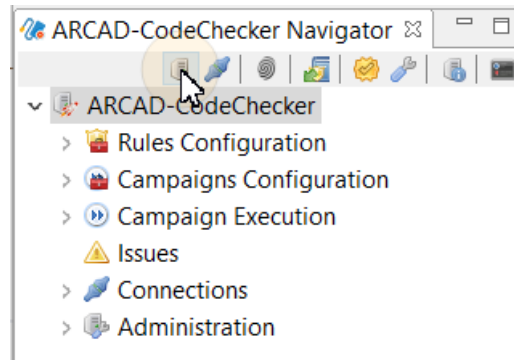


Figure 3: Configure the server connection

3.1.1 Configuring the server connection properties

The CodeChecker Studio must connect to the CodeChecker Server to access and store all the elements required for carrying out a code review process. The connection information defined when starting the studio is kept in memory in the **Connection Properties** window.

The **Connection Properties** dialog enables you to connect to another server or to use another user account. Define new connection information in the following fields:

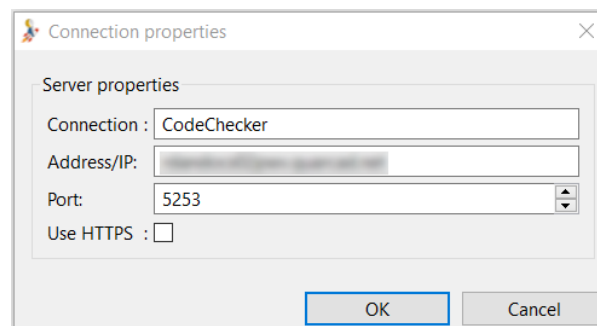


Figure 4: The server Connection Properties dialog

Connection

This field displays the default name of the CodeChecker Server connection and cannot be edited.

Address/IP & Ports

Define the URL address where the CodeChecker Server is located. This URL is usually made up of a prefix, *http://* or *https://*, then followed by the server's host name (DNS name or IP address) and the server's port. By default, the CodeChecker Server uses the 5253 port number or the 52530

port number for the secured connection, but these numbers may have been changed during configuration.

To save any changes made, click **OK**.

The CodeChecker Studio is connected to the new server. Refresh any views opened in the CodeChecker Studio to display the content associated with the new server.

Note

To connect to the server using a proxy, define the proxy parameter in the [general network connection preferences](#) of the eclipse platform (Preferences > General > Network Connections).

3.2 Connecting to the CodeChecker Server

Step 1 Click the  Connection icon in the navigator to connect to the CodeChecker Server.



Figure 5: The CodeChecker Studio connection dialog

Step 2 Define the **Login** and **Password** to connect to the CodeChecker Server.

Initially, the administration-level login must be used to define other users in the CodeChecker Studio. This default generated first user has access to all of the ARCAD CodeChecker features.

The credentials of the default ARCAD CodeChecker user are:


Login	admin
Password	quadra

Step 3 [Optional] Tick the **Retain password** box to save the connection information.

Step 4 Click **Connect**.

Result The CodeChecker Studio is connected to the CodeChecker Server and is ready to be used.

Note

To change users, click the  Connection icon in the navigator. Connect to the CodeChecker Server using a different **Username** and **Password**.

 **Reference**

For more information about configuring the CodeChecker Server, refer to [Registering license keys on page 32](#).

3.3 Configuring the TLS settings

 **Important!**

This section does not describe the configuration required for TLS in the CodeChecker Server. The TLS configuration belongs to the AFS configuration layer. Refer to the CodeChecker Server Configuration Guide.

For security reasons, you may need to use an encrypted connection between the server and the studio. To do this you must activate the use of the HTTPS protocol, by checking the **Use HTTPS** option in the server connection parameters. However, this protocol, based on the TLS network protection layer, must be correctly configured to be implemented. Depending on the server settings, the client program will have to validate the server's TLS certificate and potentially use its own key.

In most cases, only the validation of the server certificate is required. If this certificate is the product of a chain of trust whose root is a recognized Certification Authority (CA) then there is nothing special to do on the client side. The Java virtual machine already recognizes and approves certificates from all the parties in the market. However, if the server's certificate was not issued by a CA, or your organization uses a specific CA, it is necessary to add the certificate of this CA, or directly the server's certificate in the studio configuration.

It is also possible that the server has been configured in such a way that the studio must also use a key pair (private, public) that requires the server to validate the studio's certificate.

In case the server uses a certificate that is not issued by a CA or requires the studio to use a key pair, the administrator will pass these materials to you. The most secure way to do this is to send them to you in a container, called a KeyStore, which is password protected. In case the HTTPS connection requires both the validation of the server's certificate and a key pair for the studio, it is possible that the administrator will send you two stores, one containing the certificate to be approved, this is called a TrustStore, the other containing only the studio's key pair. This last store keeps the name KeyStore.

It is also possible that the administrator transmits the certificate directly to you, without integrating it into a store, in this case make sure that the source of the transmission is not spoofed. Integrating the wrong trusted certificate can jeopardize the security of the entire installation.


 **Note**

Certificates and key pairs have a limited validity period. This period varies, generally from 1 to 12 months. The following configuration procedure must therefore be repeated each time the equipment is updated. Knowing that an expired certificate or key pair will block the connection to the server.

 **Important!**

An additional security process occurs when connecting to the server with HTTPS, to verify that the connection URL domain name matches the certificate Common Name (CN). The connection to the server fails if they do not match.

Once you have the KeyStore(s) and their respective passwords, or, if not available, the server certificate, follow the steps below to redefine the HTTPS protocol settings:

Step 1 To access the **TLS Settings** dialog, right-click on the **ARCAD CodeChecker** node and select  **TLS Settings**.

Step 2 Edit the absolute paths to the **TrustStore** and **KeyStore** and their corresponding passwords.

Step 3 Click the **Import Certificate** button to import your own certificates into the file indicated in the **TrustStore** path.

You can click the **Import Certificate** button to import the server's certificate if it was sent to you as is. It will be included in the TrustStore defined in this window.




Step 4 Click **OK** to save your changes or **Cancel** to keep the default settings.

 **Note**

The **Reset** button resets access information and passwords to the default values provided by the CodeChecker Studio.

3.4 Loading default rule sets

ARCAD CodeChecker has a set of default rule sets available. The rule sets are loaded when the CodeChecker Server starts. The default rule sets contain all the configuration elements (quality rules, metrics, metric models, validation expressions) required to get started with general code quality.

You can update the default rule sets to benefit from all the latest code quality tools ARCAD offers. To load the default rule sets, right-click on the  CodeChecker Server and select the  **Reload default Rule Sets** option, or click the  Reload default Rule Sets icon in the Navigator toolbar. A dialog opens, you can choose to duplicate the ones already present (**Duplicate**), leave out the existing entities (**Only new**) or overwrite existing entities (**Overwrite**). A confirmation dialog then opens to list the number of elements successfully loaded.

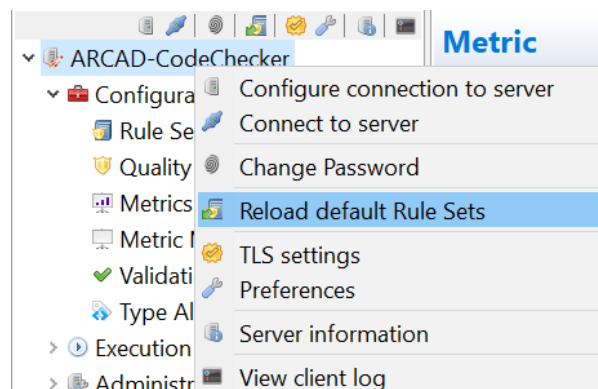


Figure 6: Reload default Rule Sets

3.5 Changing the password of a user account


Follow the subsequent steps to change the password of the currently logged-in user.

Step 1 To access the **New Password** wizard, click the  **Change Password** icon.

Step 2 Define a **New Password** and confirm it. Click **OK**.

Result The password is changed.

3.6 Verifying CodeChecker Server versions

To verify the version of a CodeChecker Server, click on the  Server Information icon in the navigator. A dialog displays the selected server's version and the user with which you are currently connected.

3.7 Viewing CodeChecker Studio logs

To view a studio's client log, click on the  View client log icon in the navigator.

4 Overview

The CodeChecker Studio is available as:

- an independent, stand-alone Eclipse studio,
- an independent perspective in RDi.

The stand-alone RCP and the RDi perspective both share the same architecture and features. They contain views and editors and control what appears in certain menus and toolbars.

To verify the version of the CodeChecker Studio, open the **Help** menu then select **About CodeChecker Studio**. A window opens to display information about the installation details and the version.

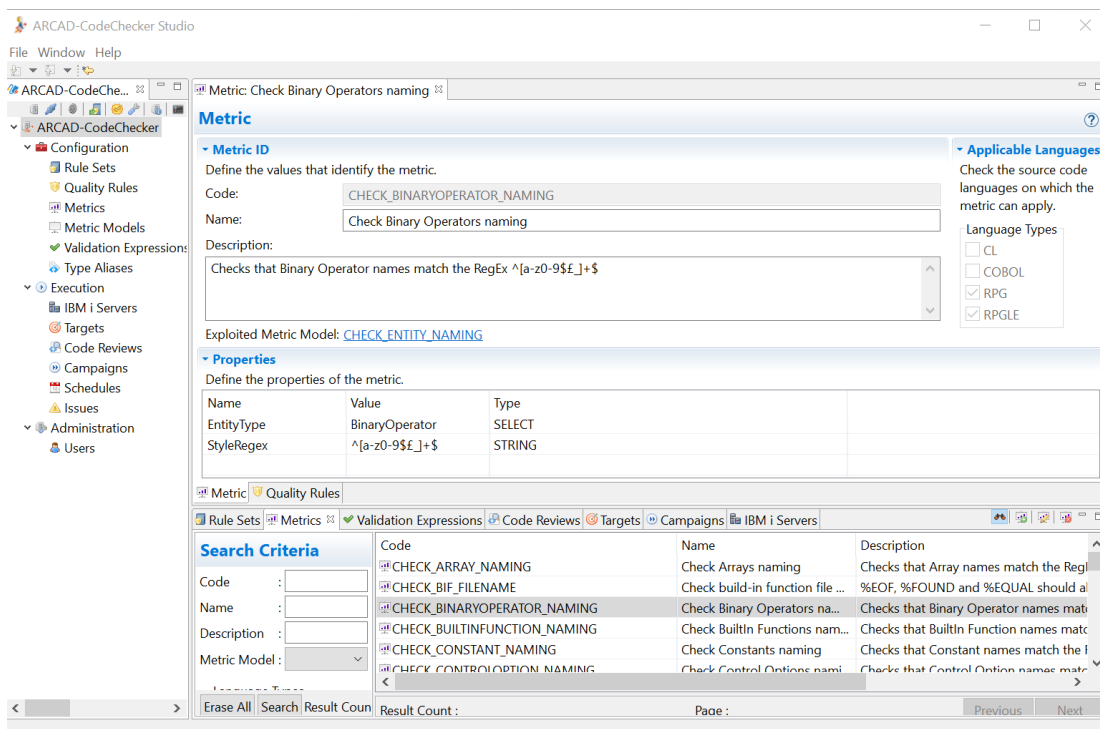


Figure 7: The CodeChecker Studio

About perspectives

A perspective defines the initial set and layout of views in the workbench window. Within the window, each perspective shares the same set of editors. Each perspective provides a set of capabilities aimed at accomplishing a specific type of task or working with specific types of resources. For example, the JavaTM perspective combines views that you would commonly use while editing Java source files, while the Debug perspective contains views that you would use while debugging a program. Perspectives contain views and editors and control what appears in certain menus and tool bars.

Views in ARCAD CodeChecker are context sensitive. To see a view that is not currently active, open it by accessing the **Show View** dialog (*Window > Show view > Other*) and selecting the view. Re-position views by dragging and dropping them to the desired location in the studio. Click an edge and drag to re-size a view. Double-click on any view's toolbar to maximize it. This is helpful when a view contains a lot of information; maximizing the view makes it fill the entire window. Double-click again to minimize the view.

To re-set the perspective to the default layout, click *Window > Reset Perspective*.

The ARCAD CodeChecker navigator

This primary view provides access to all of the elements required to set up and carry out the code review process. Double-clicking on an entity in the  **ARCAD CodeChecker Navigator** opens the corresponding view.

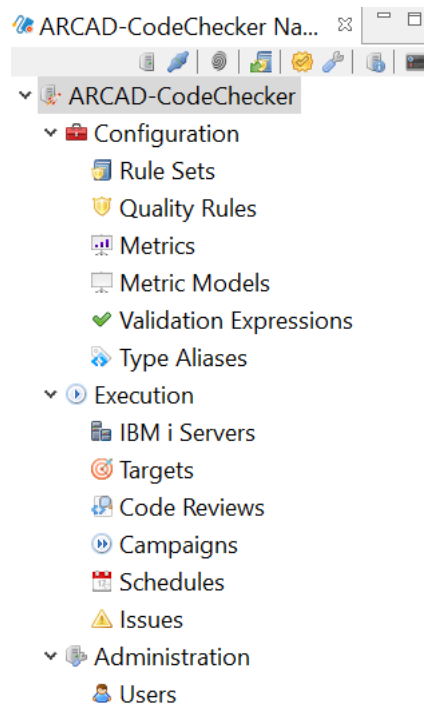


Figure 8: The ARCAD CodeChecker navigator

The administration-level user, and any other user assigned all the roles, has access to all of the entities in the navigator. A user with restricted access only has access to the entities corresponding to the roles assigned to the user profile.



CONFIGURING ARCAD CODECHECKER

Introduction to ARCAD CodeChecker configuration

This section describes how to configure the CodeChecker Studio and the CodeChecker Server.

It is not required to configure the CodeChecker Server or studio before using them. Configuration and preferences options are specific to a studio and accessible at any time to users with the appropriate role (s).

Note

Configuration and preferences options are not specific to a user account. If the CodeChecker Studio or server are configured on a workstation, the configuration options are kept the same no matter which user account is used.

5 Registering license keys

Required role	Server Administration
Access	Window Menu → Show View → Other → Configuration Categories → Administration → License Key

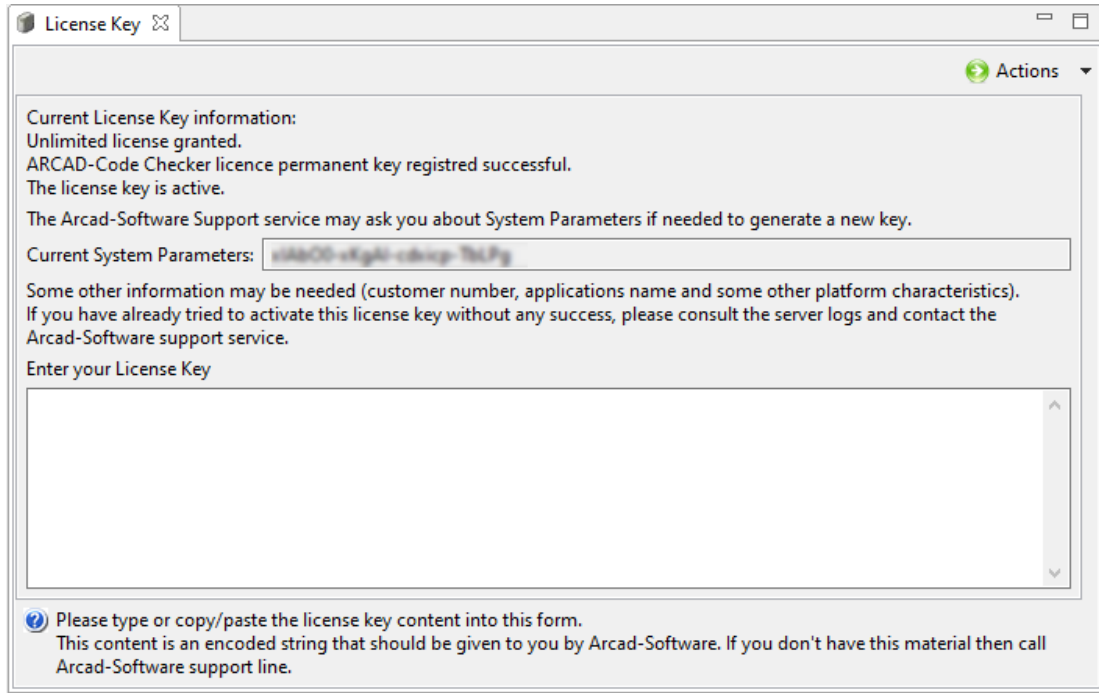


Figure 9: Register the ARCAD CodeChecker license key

Activation Keys for ARCAD CodeChecker are required to set up the ARCAD CodeChecker process. Enter the **full** License Key, including dashes, in the field provided.

Important!


Ctrl+s will not save your changes in the server configuration editors. To save, click the **Actions** drop-down menu and select **Save**.

6 Preferences

Required role	 Server Administration
Access	 Preferences

Chapter Summary

6.1 General preferences	33
6.2 Mail server preferences	34
6.3 Execution preferences	35
6.4 ARCAD Skipper precompiler preferences	35
6.5 Sonar Integration preferences	36

The preferences for ARCAD CodeChecker are managed in the  Preferences dialog. The preferences you can access depends on the Rights granted to your user profile.

Note


Preferences options are not specific to a user account. If the Studio or Server are configured on a workstation, the configuration options are kept the same no matter which user account is used.

6.1 General preferences

The **General preferences** settings section makes it possible to manage the location of all the temporary files created by the CodeChecker Server.

Temporary files root

Sets a path where the CodeChecker Server creates a **cctmp** folder dedicated to store all the temporary files or folders.

 **Example**
./temp

Note

Upon startup or when the preference is changed, the **cctmp** folder is cleared.

To save any changes made, click **Apply**. Click **Restore Defaults** to return to the default settings.

6.2 Mail server preferences

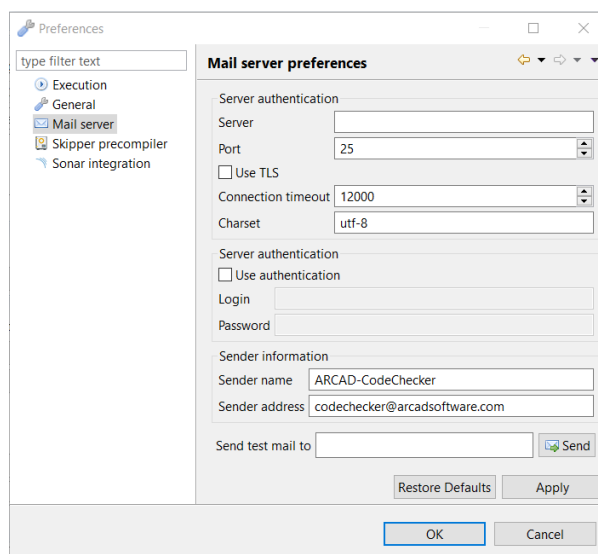


Figure 10: Mail server settings

These settings are used when an automatic email is sent, if the project allows it.

Server definition

Server: The name or the IP Address of the mail server.

Port: The communication port used to reach the mail server.

Use TLS: Tick to use TLS when establishing the connection to the mail server.

Connection Timeout: The time period in milliseconds within which the connection to the mail server must be established.

Charset: The charset used to encode non US-ASCII characters contained in the e-mail content.

Server authentication


Use Authentication: Tick to enable this option if the mail server requires authentication.

Login / Password: The login and password used to connect to the mail server.

Sender information

Sender name: The name that will be displayed in the *From* field of the sent e-mail.

Sender address: The e-mail address that will be displayed in the *From* field of the sent e-mail. This value is skipped if the **Sender name** is defined.

To test that the mail sender has been properly configured, enter an email address in the **Send test mail to** field and click the  Send button. You can put several email addresses; each address must be separated by a comma. A test email is sent to the address(es) defined. Furthermore, test emails will be sent to the address(es) every time the server is started.

Important!

If you do not want to receive an email every time the server is started, remove the address(es) entered in the **Send test mail to** field.

To save any changes made, click **Apply**. Click **Restore Defaults** to return to the default settings.

6.3 Execution preferences

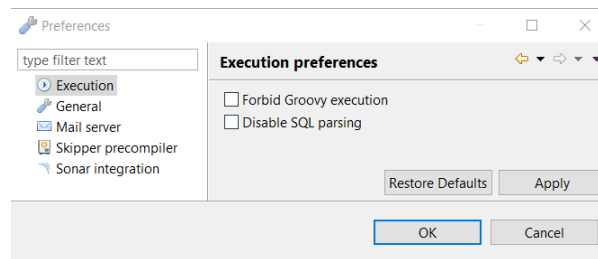


Figure 11: Execution preferences

Tick the box to **Forbid groovy execution**. Rules using groovy-based metrics will be ignored.

Tick the box to **Disable SQL parsing**. If the SQL parsing is disabled, RPG and COBOL source with SQL will still be parsed, but the resulting CAST model will not contain any SQL node. All metrics related to SQL will become irrelevant (their result will always be 0).

To save any changes made, click **Apply**. Click **Restore Defaults** to return to the default settings.

6.4 ARCAD Skipper precompiler preferences

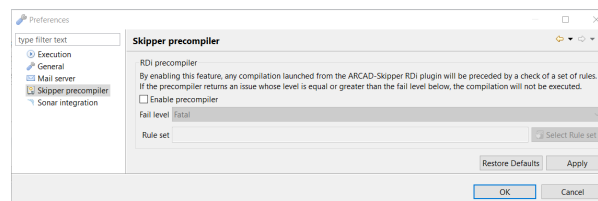




Figure 12: ARCAD Skipper precompiler settings

Tick the box to **Enable precompiler**. Any compilation launched from  ARCAD Skipper RDi plug-in will be preceded by a check of a set of rules. If the precompiler returns an issue whose level is equal or greater than the **Fail level** defined below, the compilation will not be executed. Select a  Rule set and a Fail level to configure the precompilation quality check. The rules in the rule set are executed during the precompilation.

To save any changes made, click **Apply**. Click **Restore Defaults** to return to the default settings.

Note

Once the precompiler is enabled in the ARCAD CodeChecker preferences, it is also enabled in RDi. Any compilation executed from the ARCAD Skipper RDi plug-in will execute the rules from the rule set defined in the preferences above.

If the execution fails (ie. the maximum fail level is met by one of the issues returned by the rules), a dialog informs you that the precompilation failed because the ARCAD CodeChecker rules execution failed.

The ARCAD Skipper precompiler feature can be disabled from RDi, by setting an application variable on the ARCAD Server. This specific application variable must belong to the **ARCAD_STANDARD/CODECHECKER** group, be named **PRECOMPILER** and have the value ***NO**.

Warning!

This application variable set in RDi overrides the ARCAD CodeChecker configuration, meaning that even if the **Enable precompiler** option is activated, it is not taken into account if the application variable is set with a ***NO** value in RDi.

6.5 Sonar Integration preferences

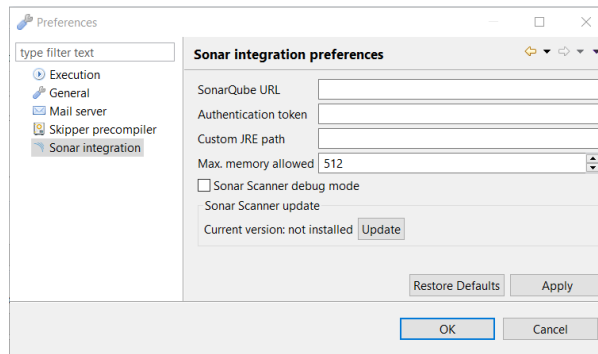


Figure 13: The SonarQube settings

The communication between the SonarQube Server and the CodeChecker Server is entirely managed by the CodeChecker Server through the following parameters:

SonarQube URL [Mandatory]

Sets the URL to the SonarQube server on which the results are uploaded.

Authentication token

Sets an authentication token used when the SonarQube server is not running on the same machine as the CodeChecker Server. The token is encrypted and stored in the ARCAD CodeChecker database.

Custom JRE path

Sets a JRE that can be used by putting its installation path here. By default, the Sonar Scanner is executed by using the current Java Runtime Environment.

Max. memory allowed

Sets the maximum memory allowed to the Sonar Scanner to perform its operation.

In case of a **Java heap space error** or a **java.lang.OutOfMemoryError** happening during the execution, the maximum allowed memory can be increased here.

Sonar Scanner debug mode

Enables or disables the debug mode for the Sonar Scanner.

Sonar Scanner update

Current version: Indicates the version of Sonar Scanner currently installed.

If no Sonar Scanner is installed, the status is set to **not installed**.

Click the **Update** button to update automatically the Sonar Scanner when it is already installed.

If the latest version is already installed, a warning dialog opens to let you know the Sonar Scanner cannot be updated, as it is installed already in its latest version.

 **Reference**

For more information about the installation instruction for the Sonar Scanner, refer to the ARCAD CodeChecker Installation Guide.



USER ADMINISTRATION

Introduction to user administration

ARCAD CodeChecker administrators are in charge of managing users and roles. It is important to define your team and associate the correct roles with each user, because the individuals who use ARCAD CodeChecker at different stages in the process must have accurate authority levels in order to access and, in some cases, create and/or edit the various processes.

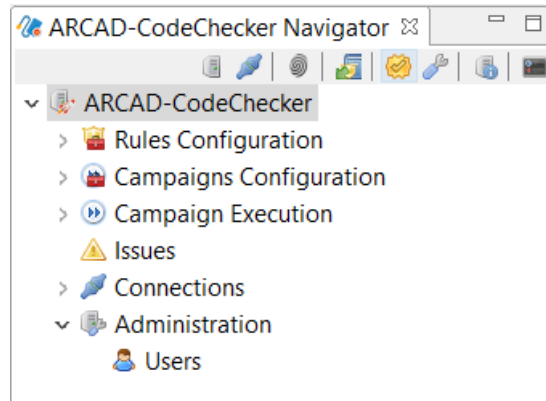






Figure 14: The Administration node in the navigator

User information can be stored locally in ARCAD CodeChecker and the login credentials for each user are managed by an administrator or any user with the corresponding role. Any user can change his or her own password in the navigator.

This section covers all of an administrator's tasks associated with:

-  [Users on page 40](#)
-  [User roles on page 44](#)

7 Users

Required role	 User Management
Access	 CodeChecker Server →  Administration →  Users

Chapter Summary

7.1 Creating users.....	40
7.2 Importing users.....	41
7.3 Editing users.....	42
7.4 Deleting users.....	43

User accounts define access rights for each ARCAD CodeChecker user and are managed in the CodeChecker Studio. Each user account has a unique login.

It is important to define users and assign the correct roles to each one. The individuals who use ARCAD CodeChecker at different stages during the analysis of an application's code quality must have accurate authority levels in order to access the various processes.

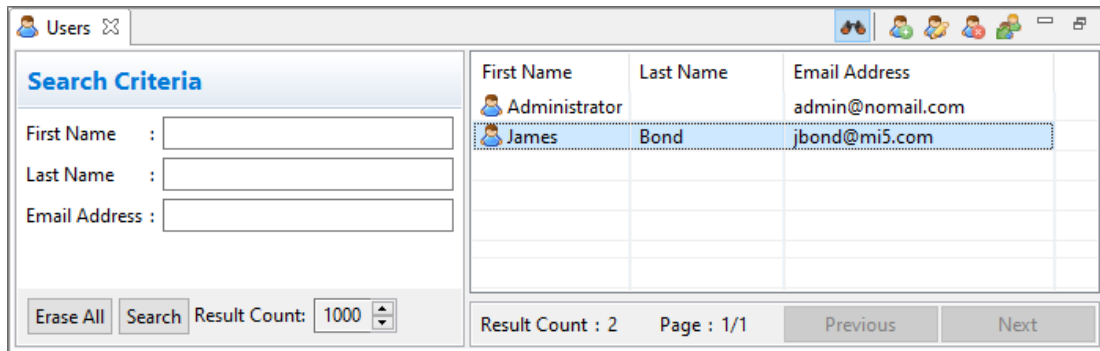




Figure 15: User Management

The  **Users** search view is accessed from the  **Administration** node in the **Navigator**.

Enter any combination of the above search criteria, then click the **Search** button to display the results. To display the complete list, click the **Search** button without entering any search criteria.

7.1 Creating users

Users created in the studio store their connection details on the CodeChecker Server.

Follow the subsequent steps to create a new user.

Step 1 To access the **New User** wizard, click the  Create icon in the  **Users** view.

Step 2 Define the user's ID. It is required to define either a **First Name** or a **Last Name** to create a new user, but all the following values can be edited later.

- Select the **Title** from the drop-down list.
- Define the user's **First Name**, **Last Name** and **Email Address**.


Click **Next >** to continue.

Step 3 Define the user's login details. It is required to define the user's login details to create a new user, but all the fields can be edited later.


- Enter the user's [Login](#) and [Password](#).
- Select a password validity limit. By default, the password is valid for a year after the day of creation.
- Tick the box to lock the user account if needed. The user will be created but the account will be deactivated.


Click **Next >** to continue.

Step 4 Select the user's role(s). It is possible to leave all the roles unchecked, or to check as many roles as required.

 **Reference**
For more information about user roles, refer to [User roles on page 44](#).

Click **Finish**.

Result The new user is created and is displayed in the  **Users** view.

 **Important!**
You can always edit the user information by double-clicking on the user.


7.2 Importing users


Users can be imported from an IBM i or LDAP login source depending on the server configuration. Follow the subsequent steps to import users.

Step 1 To access the **Import Users** wizard, click the  **Import Users** icon in the  **Users** view.

Step 2 Select the login source from which to import the users.

Tick the **Use a user directory (LDAP)** or the **Use an IBM i login**.

 **Important!**
The login source selection boxes are not displayed if the login sources are not already configured.

 **Reference**
For more information about configuring the connection to external login sources like an LDAP or IBM i users directories, refer to the CodeChecker Server Configuration Guide.



Click **Next >** to continue.

Step 3 Select the user(s) to import and click **Next**.

Step 4 Define the role(s) to assign to all of the imported users. It is possible to leave all the roles unchecked, or to check as many roles as required. Click **Finish**.

Result The new user(s) is/are imported and displayed in the  **Users** view.

7.3 Editing users

To edit a user, either select it in the  **Users** view and click the  Edit icon, or double-click on a user in the view. The editor is displayed in a separate view.

Save the changes (, Ctrl+S or **File > Save**).

User Identification

Title

Select a title from the drop-down list.

First Name

Define the user's first name.

Last Name

Define the user's last name.

Email Address

[Optional] Define the user's email address. This address is only used for informational purposes.

Local Authentication

Login

This login is the user's ID and must be unique. Paired with the [Password](#), it enables users to connect to the CodeChecker Server.

Password

Define a password, then click **Test** to make sure the password is valid. Paired with the [Login](#), it enables users to connect to the CodeChecker Server.

Specific conditions can be configured for passwords. The password must then comply with all the conditions set to be valid.



Reference

For more information about configuring password conditions, refer to the CodeChecker ServerConfiguration Guide.

Password Confirmation

The [Password](#) defined must be entered a second time for confirmation. The password confirmation must be identical to the password.

Password Validity Limit

Define an expiration date for the password. When this date is reached, the user will have to change their password to access the account.



Reference

For more information about changing the password of a user account, refer to [Changing the password of a user account on page 26](#).

Lock the user account

Uncheck this box to unlock a user account.

An account is locked after a number of unsuccessful attempts to log in. The access to the user account is locked, meaning the user cannot use its connection information to connect to the CodeChecker Server.

IBM i Authentication

IBM i Profile

Enter the login of the IBM i user account, then click **Check** to make sure the login is valid. The password associated with the IBM i login is retrieved automatically.

LDAP Authentication

LDAP user

Enter the login of the LDAP user account, then click **Check** to make sure the login is valid. The password associated with the LDAP login is retrieved automatically.

Roles

This section enables you to manage the roles assigned to user profiles.

Depending on the roles assigned, a user has access to specific features of the CodeChecker Studio. To prevent users from accessing any of the studio's features, do not select any roles.



Reference

For more information about user roles, refer to [User roles on page 44](#).

7.4 Deleting users

Warning!

Deleted users cannot be recovered.

To delete a user, select it in the  **Users** view and click the  Delete icon. Click **OK** to confirm or click **Cancel** to keep the user.

8 User roles

User roles make it possible to allow or deny access to specific features of the CodeChecker Studio.


























Role	Description
 Server Administration	Enables users to configure the CodeChecker Server, and to access, create and manage users under the  Administration node in the navigator.
 Connection Management	Enables users to access, create and manage all the entities available under the  Connection node in the navigator: <ul style="list-style-type: none">  IBM i Servers  SSH Keys
 Rule Management	Enables users to access, create and manage all the entities available under the  Rule Configuration node in the navigator: <ul style="list-style-type: none">  Rule Sets  Quality Rules  Metrics  Metric Models  Validation Expressions <p>This role includes the action to activate or deactivate rule sets and to activate quality rules.</p>
 Campaign Management	Enables users to access, create and manage all the entities available under the  Campaign Configuration node in the navigator: <ul style="list-style-type: none">  Type Aliases  Targets  Code Reviews
 Campaign Execution	Enables users to access, create and manage all the entities available under the  Campaign Execution node in the navigator: <ul style="list-style-type: none">  Campaigns  Schedules
 Issues Management	Enables users to access, create and manage all the entities available under the  Issues node in the navigator: <ul style="list-style-type: none">  Issues

Table 4: The user roles



CONNECTIONS

Introduction to connections

Create and manage connections to your IBM i server as objects in the CodeChecker Server, to use them when configuring the targets of your code quality campaigns. You can configure the connection to any IBM i server, with or without ARCAD. Credentials are safely stored and you can test immediately the communication between ARCAD CodeChecker and your IBM i server.

-  [IBM i servers on page 47](#)

The SSH keys used for Git connections are also managed in the connections.

-  [SSH Keys on page 50](#)

9 IBM i servers

Required role	Connection Management
Access	CodeChecker Server → Connections → IBM i Servers

Chapter Summary

9.1 Creating IBM i server connections	47
9.2 Set a default IBM i server.....	48
9.3 Deleting IBM i server connections	49

IBM i servers define the connections to the machines where the code to be reviewed is managed. When you create a connection to an IBM i server the connection details are stored by ARCAD CodeChecker and can be used when defining Targets.

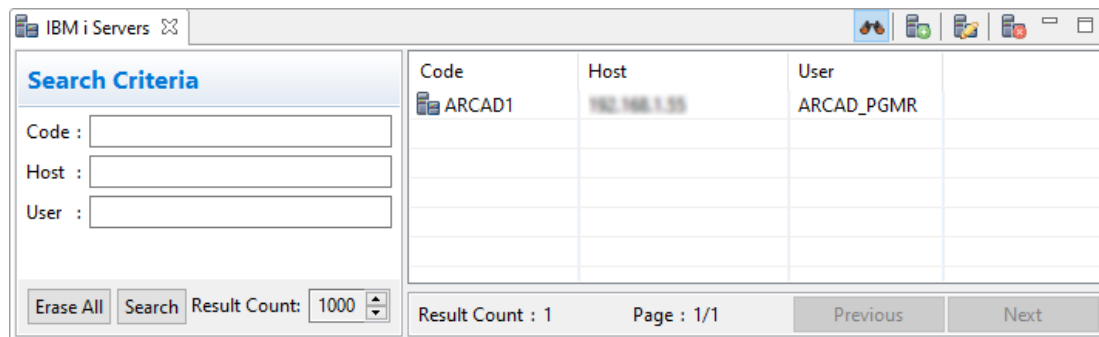


Figure 16: IBM i Servers

The **IBM i Servers** view is accessed from the **Connections** node in the **Navigator**.

Enter any combination of the above search criteria, then click the **Search** button to display the results. To display the complete list, click the **Search** button without entering any search criteria.

9.1 Creating IBM i server connections

Follow the subsequent steps to create a new IBM i server connection.

Step 1 To access the **Create a new IBM i server** wizard, either click the Create icon in the toolbar of the **IBM i Servers** search view, or right-click anywhere in the list and select **Create IBM i server**.

Step 2 Define the following connection information:

Code

Enter a unique name to identify this connection to the server.

Host

Enter the name or IP address of the IBM i partition hosting the server.

User/Password

Enter a valid user login and password. This user must be declared on the server and will be the user that will execute any operations on the IBM i. The password should be encoded in Base64.

Use SSL

Tick this checkbox if the server requires a secured TLS/SSL connection.



Reference

For more information about ARCAD secure connections, refer to the *ARCAD SSL Configuration Guide*.

Auto CCSID

Tick this checkbox to automatically set the CCSID. The CCSID will be determined by the server once it connects.

CCSID


Enter a valid CCSID to connect to the server.

Default sever

Tick this checkbox to automatically set this server as the default IBM i server.

Step 3 Click the **Test connection** button to check if the IBM i server is connected.

Step 4 Click **Finish**.

Result The IBM i server is defined and ready to be used in  Targets.

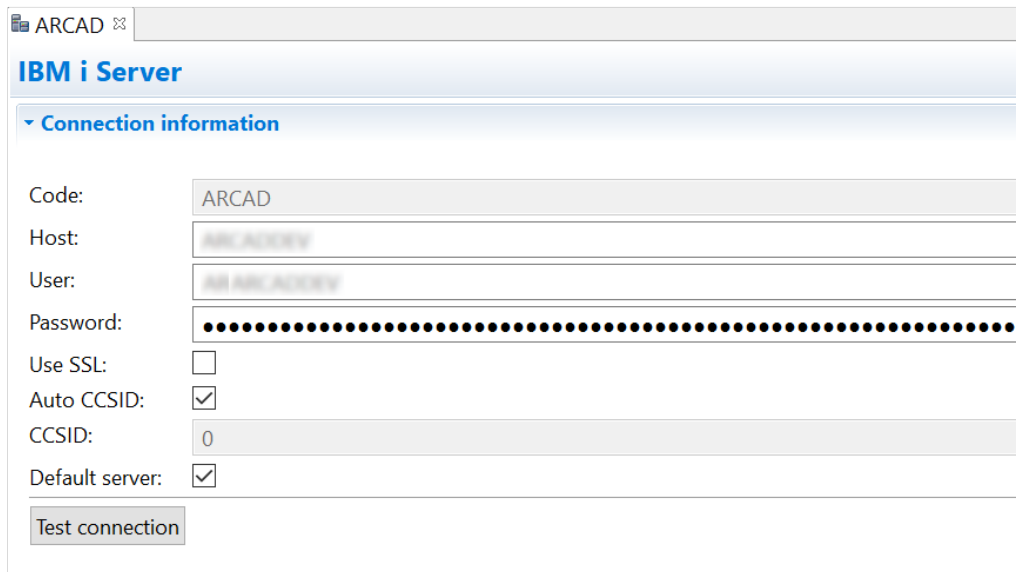




Figure 17: The IBM i server editor

Connections to target servers are editable. To edit an existing connection, click the  Edit icon. Any modification is immediately taken into account in the targets using the IBM i server connection.

9.2 Set a default IBM i server

There is now a default IBM i server set in ARCAD CodeChecker. If there is no default server at the CodeChecker Server start, it is the IBM i server with the smallest ID in database that automatically




becomes the default IBM i server.

Setting a default IBM i server means that it is automatically used when creating a  Target, without having to specify a server in the process.

 **Warning!**

It is mandatory to have an IBM i server set as default and there can be only one default IBM i server.

To set an IBM i server as the default one in ARCAD CodeChecker, you can either:

- Right-click on a server from the  **IBM i Servers** view, then click the  **Set default IBM i Server** option.
- Tick the **Default server** checkbox from the  IBM i server editor.

 **Note**




If a default IBM i server is deleted, it is the server with the smallest ID remaining in the list that replaces it.

If another IBM i server is marked as the default one, the previous one loses its default attribute.





9.3 Deleting IBM i server connections

 **Warning!**

Deleted IBM i server connections cannot be recovered.

To delete an IBM i server connection, either right-click on it in the  **IBM i Servers** search view and select  **Delete**, or select it and click the  Delete icon in the toolbar. Click **OK** to confirm or click **Cancel** to keep the connection.

10 SSH Keys



Required role	 Connection Management
Access	 CodeChecker Server →  Connections →  SSH Keys

Chapter Summary



10.1 Creating SSH keys.....	50
10.2 Editing SSH keys.....	51
10.3 Accessing the public key.....	51
10.4 Deleting SSH keys.....	52


ARCAD CodeChecker can generate SSH key pairs to facilitate secured connections to external target tools. It is not required to use an SSH key generated by ARCAD CodeChecker to connect to external tools but it is an option among others.

An SSH key pair is comprised of:

-  the private key, referenced in ARCAD CodeChecker.
-  the public key, which must be referenced by the system to which ARCAD CodeChecker needs to connect.

The SSH keys generated by ARCAD CodeChecker are either EdDSA or RSA keys.


The  **SSH Keys** view is accessed from the  **Connections** node in the **Navigator**.

The  **SSH Keys** view displays all of the keys defined for the current server. Each key's fingerprint is displayed in the Fingerprint column.

10.1 Creating SSH keys



There are multiple ways to create SSH keys in ARCAD CodeChecker:

- [Create new SSH keys below](#)
- [Import existing SSH keys on the facing page](#)

 **Important!**
ARCAD encourages you to create or import **EdDSA keys**.
Most of the Git providers support EdDSA keys, as they are more secure and consumes less resources to be generated and used. Furthermore, RSA keys and their support are likely to be abandoned in the future.

10.1.1 Create new SSH keys

Follow the subsequent steps to create a new SSH key pair.

Step 1 To access the **SSH Key** wizard, either click the  create icon in the **SSH Keys** search view or right-click anywhere in the view and select  **Create new SSH key**.

Step 2 Enter a recognizable ID in the **Key Identifier** field.

Step 3 Select the **Type** of key to generate. You can choose between an EdDSA key and a RSA key.

Step 4 Enter the passphrase for the SSH key pair.

Note

The passphrase will not be displayed in order to ensure absolute confidentiality. Be sure to enter the correct passphrase and not to forget it.

Step 5 Click **Finish** to create the new key. The new key appears in the list.



Result SSH keys are saved in the CodeChecker Server installation directory in the following path:
`../ssh/keystore/ks<n>` (where n is the key's internal ID).

There are two files, each representing half of the key pair:

- the private part of the key which ARCAD CodeChecker will reference.
- the public part of the key which must be referenced by the target tool.

10.1.2 Import existing SSH keys

Follow the subsequent steps to import an existing SSH key. Only the private half of the SSH is imported into ARCAD CodeChecker.

Step 1 To import an existing private key, either right-click anywhere in the view and select  **Import SSH key** or click the  import icon.

Step 2 Enter a recognizable ID in the **Key Identifier** field.



Step 3 Browse to the location where the existing private key file is stored and select it in the **Private key** field.

Step 4 Enter the passphrase for the key if one was defined.



Step 5 Click **Finish**.

Result The private key is saved in the CodeChecker Server installation directory in the following path:
`../ssh/keystore/ks<n>` (where n is the key's internal ID).

10.2 Editing SSH keys

To view or edit a key's ID, either right-click on it and select  **Edit**, select the key then click the  Edit icon or double-click on it. The **SSH Key** editor is opened where its ID can be edited. You can also add comments in the SSH key editor but you cannot edit the key pair.



10.3 Accessing the public key

To display the public key, either right-click on the SSH key and select  **Show Public Key** or select the key in the list and click the  public key icon. You can copy the public key from the dialog and use it to create the SSH connection in the target tool.

10.4 Deleting SSH keys

 **Warning!**

Deleted SSH Keys cannot be accessed or recovered and will no longer be useable. If a referenced SSH key is deleted, the entity referencing it will no long be able to connect to its target.

To delete an SSH key either right-click on it and select  **Delete** or select it and click the  Delete icon. Click **OK** to confirm or click **Cancel** to keep the key.



QUALITY RULES CONFIGURATION

Introduction to quality rules

Quality rules are the base of code quality. The rule configuration is the quality policy you want to apply to your development, a set of conditions against which your code is measured.

ARCAD CodeChecker supplies a set of standard rules that reflect common quality standard and industry best practices, that you can use out of the box. However, you can set your own standard by creating your own rules. The rule sets provided are rules grouped by theme reflecting a quality goal to achieve. It is the rule sets that are called to define the scope of the [Code reviews](#).

ARCAD CodeChecker makes it easy to set up the quality rules, the metrics and all the other entities to use for measuring the code quality of an application. These entities must be configured before defining code reviews or creating campaigns.

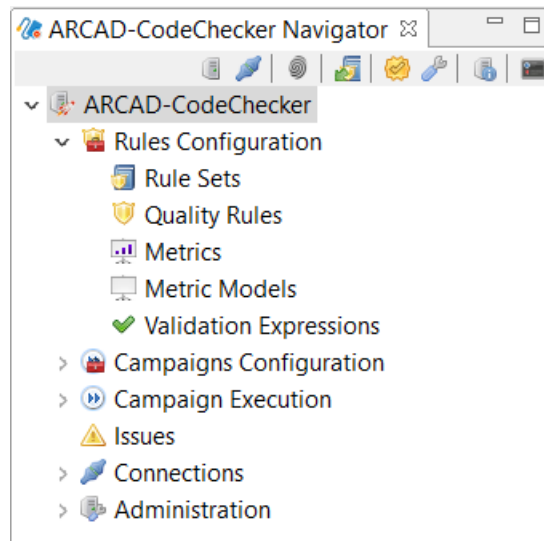


Figure 18: The Rules Configuration node in the navigator

All of the following entities are accessed and managed in the **Rules Configuration** node in the navigator, under the currently connected **CodeChecker Server**.

- [Rule sets on page 55](#)
- [Quality rules on page 62](#)
- [Metrics on page 69](#)
- [Metric models on page 73](#)
- [Validation expressions on page 78](#)

11 Rule sets

Required role	Rule Management
Access	CodeChecker Server → Rules Configuration → Rule Sets

Chapter Summary

11.1 Creating rule sets	56
11.2 Editing rule sets	56
11.3 Activating rule sets	58
11.4 Loading default rule sets	59
11.5 Open the rule set documentation	59
11.6 Exporting and importing rule sets	60
11.7 Deleting rule sets	61

The [Quality rules](#) used to analyze the source code of an application are grouped into rule sets in the CodeChecker Studio.

Use rule sets to organize quality rules in a logical way: by language, by theme, and so on. The standard quality rules are already organized into several generic rule sets. You can create new rule sets to manage and categorize your custom quality rules, or re-organize the standard quality rules to fit your needs.

Important!

A rule set must be activated to be used.

Note

ARCAD CodeChecker supplies a number of standard, ready-to-use rule sets, that include all the elements required for their configuration (quality rules, metrics and validation expressions)

Reference

For more information about individual quality rules, refer to [Quality rules on page 62](#).

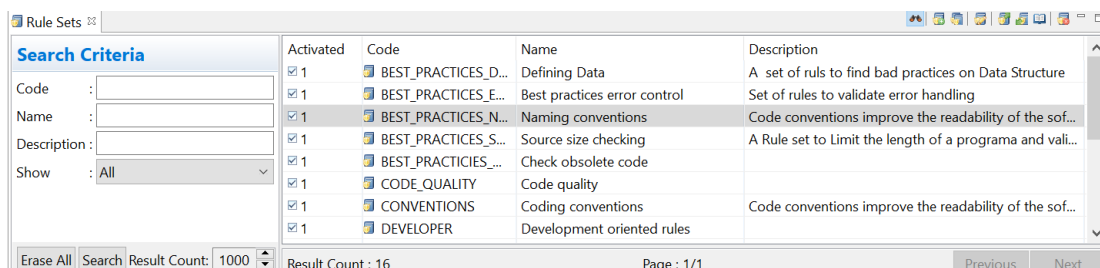





Figure 19: Rule Sets



The **Rule Sets** search view is accessed from the **Configuration** node in the **Navigator**.

Enter any combination of the above search criteria, then click the **Search** button to display the results. To display the complete list, click the **Search** button without entering any search criteria.

11.1 Creating rule sets

Follow the subsequent steps to create a new rule set.


Step 1 To access the **Create Rule Set** wizard, either click the  Create icon in the toolbar of the  **Rule Sets** search view, or right-click anywhere in the list of the view and select  **Create Rule Set**.

You can also create a new rule set by right-clicking on an existing one and selecting the  **Duplicate Rule Set** option, or select a rule set and click the  Duplicate Rule Set icon in the toolbar.

Step 2 Define the rule set's **Code** and **Name**. These values are required to create a new rule set but can be edited later.

If the new rule set is created by duplicating an existing one, edit its **Code** and **Name** values.


Click **Finish**.



Result The new rule set is created and is activated automatically. It is displayed in the search list in the  **Rule Sets** search view.



 **Important!**

New sets do not contain any quality rules yet and must be edited before they can be used.

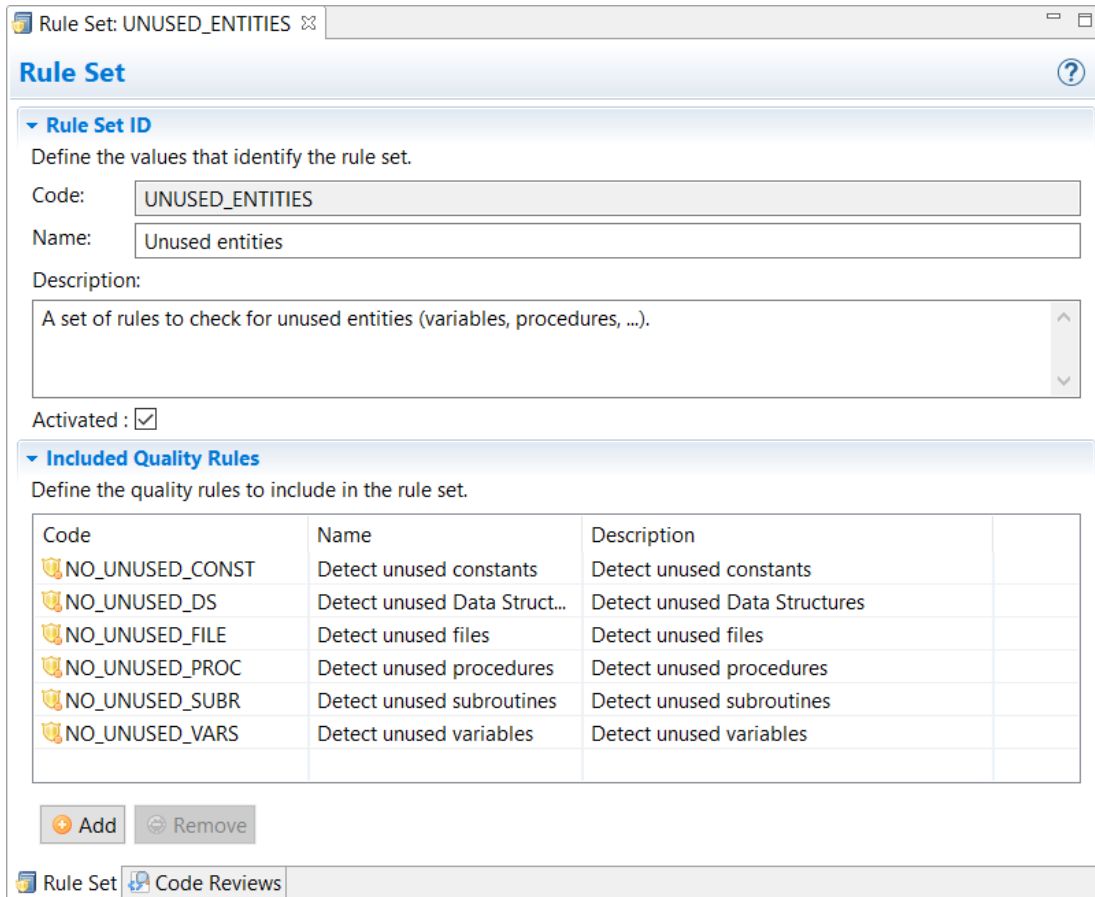
11.2 Editing rule sets

To open a rule set's editor, locate it in the  **Rule Sets** search view, then either:

- double-click on the item in the view,
- right-click on the item in the view and select  **Edit**, or
- select the item in the view and click the  Edit icon in the toolbar.

The editable values are all located in the  **Rule Set** tab in the rule set's editor. The  **Code Reviews** tab displays the list of code reviews that include this rule set. Double-click on a code review to access its editor.

Save the changes (, Ctrl+S or **File > Save**).



The screenshot shows the 'Rule Set' editor for 'UNUSED_ENTITIES'. It includes fields for 'Code' (UNUSED_ENTITIES), 'Name' (Unused entities), and 'Description' (A set of rules to check for unused entities (variables, procedures, ...)). There is an 'Activated' checkbox which is checked. Below, a table lists 'Included Quality Rules' with columns for Code, Name, and Description. At the bottom, there are 'Add' and 'Remove' buttons and a breadcrumb trail showing 'Rule Set' and 'Code Reviews'.

Code	Name	Description
NO_UNUSED_CONST	Detect unused constants	Detect unused constants
NO_UNUSED_DS	Detect unused Data Struct...	Detect unused Data Structures
NO_UNUSED_FILE	Detect unused files	Detect unused files
NO_UNUSED_PROC	Detect unused procedures	Detect unused procedures
NO_UNUSED_SUBR	Detect unused subroutines	Detect unused subroutines
NO_UNUSED_VARS	Detect unused variables	Detect unused variables

Figure 20: The Rule Set editor

Rule Set ID

Code

This code is the rule set's ID. The code can be the same value as the [Name](#) but must be unique among other rule sets.

Name

The rule set's name should reflect a category or the type of quality rules that are contained in it to be easily identified.

Description

[*Optional*] Enter a detailed description of the rule set. This description is displayed in the ARCAD CodeChecker plug-in for RDi to provide additional information.

Activated

When this box is checked, the rule set. A rule set must be active to be used in a code review or in the ARCAD CodeChecker plug-in for RDi.




Reference

For more information about activating rule sets, refer to [Activating rule sets on the next page](#).

Included Quality Rules


This section enables you to manage the quality rules included in the rule set.





Follow the subsequent steps to add quality rules to a rule set.

Step 1 Open the desired rule set in the  **Rule Sets** view.

Step 2 Click the  **Add** button. The  **Select Quality Rules** wizard opens.


Select one or several [Ctrl+click] quality rule(s) from the list.

To filter the list of quality rules displayed in the  **Select Quality Rules** dialog and make finding the correct quality rule easier, use the **Filter** field at the top of the dialog. You can also use the tick-boxes to restrict the rule search to a specific language. Both filtering methods can be combined. The list of quality rules is updated automatically to match the defined filters.

You can also copy and paste rules from one rule set to another, or directly from the  Quality Rules view, by right-clicking on a rule and selecting the  **Copy** or  **Cut** option. You can then open another rule set, right-click and select the  **Paste** option. You can use the following keyboard shortcuts: Ctrl + C, Ctrl + X and Ctrl + V.

Click **OK**.




Important!

Only active quality rules can be added to a rule set. The quality rules that are not yet activated will not be displayed in the  **Select Quality Rules** wizard.

Reference

For more information about activating quality rules, refer to [Activating quality rules on page 67](#).

Result The selected quality rules are added to the rule set.

To remove a quality rule from a rule set, either right-click on the quality rule in the list in the  **Rule Set** editor and select  **Remove**, or select the rule(s) [Ctrl+click] and click the  **Remove** button.

11.3 Activating rule sets

Required role

<input checked="" type="checkbox"/> Activation

Activating a rule set means this rule set can be used in a code review or in the ARCAD CodeChecker plugin for RDi.

To activate a rule set, check **Activated** in the rule set's editor.

If a rule set is not active, it is considered deactivated and cannot be used. The quality rules included in a deactivated rule set cannot be executed either.

Note

Single quality rules can also be activated or deactivated. Deactivated rules, even if they are included in an activated rule set, cannot be executed either.

11.4 Loading default rule sets

ARCAD CodeChecker has a set of default rule sets available. The rule sets are loaded when the CodeChecker Server starts. The default rule sets contain all the configuration elements (quality rules, metrics, metric models, validation expressions) required to get started with general code quality.

You can update the default rule sets to benefit from all the latest code quality tools ARCAD offers. To load the default rule sets, right-click on the CodeChecker Server and select the **Reload default Rule Sets** option, or click the Reload default Rule Sets icon in the Navigator toolbar. A dialog opens, you can choose to duplicate the ones already present (**Duplicate**), leave out the existing entities (**Only new**) or overwrite existing entities (**Overwrite**). A confirmation dialog then opens to list the number of elements successfully loaded.

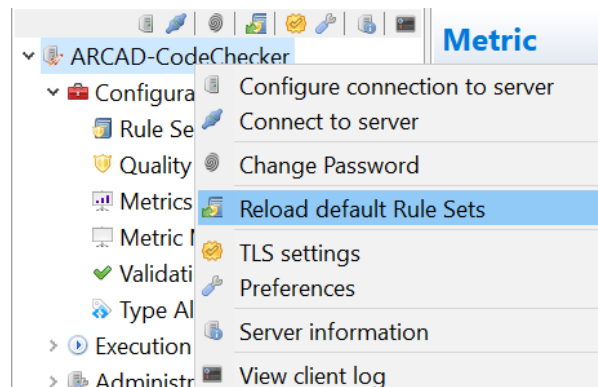


Figure 21: Reload default Rule Sets

11.5 Open the rule set documentation

To open the rule sets documentation in a web page, click on the Documentation icon in the toolbar. You can also right-click anywhere in the list of rule sets and select the **Open the Rule Sets documentation** option.

The documentation for all the rules is displayed in a web-browser, including your custom rules and rule sets.

ARCAD-CodeChecker rule sets

Code quality

- Code quality
- Binary operators with identical expressions on both sides not allowed
- Block nesting limit = 6
- Blocks can't be empty
- Calls must have prototype
- Comments limit = 300 characters
- Conditions blocks limit = 20 lines
- Detect if blocks with more than 2 Elseif
- Detect #/Elseif blocks with no default
- Detect complex procedures
- Detect even lengthed non packed fields
- Detect indicators not compared with figurative constants
- Detect lengthy NOMAIN modules
- Detect procedures with no return/output parameter
- Detect source member sequences inconsistency
- Detect sources not using *SRCSTMT
- Detect unchecked (e) extended operations
- Detect uninitialized *PSSR calls
- Detect usage of POS keyword for DS subfields
- Detect variables self-assignment
- Duplicated literals must be replaced by constant
- Exported procedures only in NOMAIN module
- Expressions are limited to 4 OR/AND
- Find duplicated file/objects
- Forbid CALLS without PLIST
- Forbid DUMP
- Forbid GOTO statements
- Forbid Select *
- Forbid debug control specification
- Forbid indicator conditioned ifs
- Forbid non keyed file access
- Global variable only used in 1 sub-procedure not allowed
- Multi-occurrence DS are obsolete
- No duplicated conditions
- No global variable in subprocedure
- Operation code END is obsolete
- Subroutines must be documented
- Use figurative constant in expression when possible
- WORKSTN files must have INDDS

Files declaration/usage

- Check file error handling
- Remove PDB declaration and avoid of link = DIC

Binary operators with identical expressions on both sides not allowed

Detect binary operators with the same expression on both sides.

Non compliant code

```
if w_flg1 = w_flg2 // always true
  dply w_flg1;
endif;

if w_flg1 < w_flg1 // always false
  dply w_flg1;
endif;

if w_flg1 = w_flg1 and w_flg2 = w_flg2 //first is true then always true
  dply w_flg1;
endif;

if w_flg1 = w_flg1 or w_flg2 = w_flg2 //first is true then always true
  dply w_flg1;
endif;
```

Compliant code

```
if w_flg1 = w_flg2 ;
  dply w_flg1;
endif;

if w_flg1 < w_flg2 ;
  dply w_flg1;
endif;

if w_flg1 = w_flg2 and w_flg2 = w_flg2 ;
  dply w_flg1;
endif;

if w_flg1 = w_flg2 or w_flg2 = w_flg2 ;
  dply w_flg1;
endif;
```

Block nesting limit = 6

Having less nested loops ensures simpler code. Simplified code is always easy to understand and maintain for future enhancements.

Non compliant code

```
if (ISROUND(OLDCUSTOMP));
  Exec SQL
  select NAME into :w_Name
  from CUSTOMERP
  where ID = :wCustomerId;

  if SQLCODE <> '0';
    Exec SQL
    select DELETED into :w_Deleted
    from TRASH
    where ID = :wCustomerId;

  if SQLCODE = '0';
    if LoggerAvailable() = 'NO';
      if systemName = 'PRODUCTION';
        Logger('Inconsistency in customer ID ' + wCustomerId);
      endif;
    endif;
  endif;
endif;
```

Compliant code

```
if (ISROUND(OLDCUSTOMP));
  Exec SQL
  select NAME into :w_Name
  from CUSTOMERP
  where ID = :wCustomerId;

  if SQLCODE <> '0';
    Exec SQL
    select DELETED into :w_Deleted
    from TRASH
    where ID = :wCustomerId;

    if SQLCODE = '0' AND LoggerAvailable() = 'NO' AND systemName = 'PRODUCTION';
      Logger('Inconsistency on customer ID ' + wCustomerId);
    endif;
  endif;
endif;
```

Figure 22: The rule sets documentation

11.6 Exporting and importing rule sets

Exporting and importing rule sets is useful when sharing configurations between multiple servers or users.

A rule set's exported configuration includes its dependent entities as well as the properties and definitions associated with these entities, maintaining the relationships between them.

All the original configuration that came from the exported entities are retrieved when importing the rule set again in ARCAD CodeChecker.

An exported rule set may include:




- quality rules,
- metrics,
- metric models,
- validation expressions.

Note

The CodeChecker Server comes with default rule sets to populate all the default configuration elements (rule sets, metrics, etc.).

11.6.1 Exporting rule sets

Follow the subsequent steps to export a rule set to a *.json* file.

Step 1 Select the rule set(s) to export. Click the  Export icon in the toolbar of the  **Rule Sets** search view, or right-click and select  **Export** in the menu.

Step 2 Select the location in which to save the exported file(s) and change the file name(s), if needed.

By default, rule sets are exported with the file name: *ARCAD_CodeChecker.json*.

Click **Save**.

Result The rule set's configuration is available externally and can be imported as such for other ARCAD CodeChecker servers or users.

11.6.2 Importing rule sets

Follow the subsequent steps to import a rule set from a *.json* file.

Step 1 Click the  Import icon in the  **Rule Sets** search view, or right-click anywhere in the view and select  **Import**.

Step 2 Select the import file. Click **Open**.

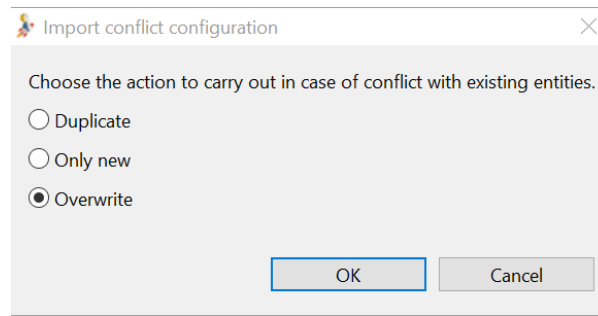


Figure 23: The Rule Set import dialog

ARCAD CodeChecker detects the items on the server that share the same code (ID) as the one from the import file.

Step 3 Depending on the content of the import, you may choose to duplicate the ones already present (**Duplicate**), leave out the existing entities (**Only new**) or overwrite existing entities (**Overwrite**).

- Duplicate: the conflicting objects are imported as duplicates of existing objects with different codes.
- Only new: the conflicting objects are ignored (new objects are imported).
- Overwrite: the conflicting objects are imported (existing objects are replaced).

Result The imported rule set(s) and related entities are displayed in their respective search views.

11.7 Deleting rule sets

Warning!

Deleted rule sets cannot be accessed or recovered.

To delete a rule set, either right-click on it in the  **Rule Sets** search view and select  **Delete**, or select it and click the  Delete icon in the toolbar. Click **OK** to confirm or click **Cancel** to keep the rule set.

12 Quality rules

Required role	 Rule Management
Access	 CodeChecker Server →  Rules Configuration →  Quality Rules

Chapter Summary


- 12.1 Creating quality rules..... 62
- 12.2 Editing quality rules..... 63
- 12.3 Quality rules batch update..... 66
- 12.4 Documenting quality rules..... 66
- 12.5 Activating quality rules..... 67
- 12.6 Adding quality rules to a rule set..... 68
- 12.7 Deleting quality rules..... 68

Quality rules are conditional expressions used to determine an application's code quality.

A quality rule is built around a metric and a validation expression. A metric is a numeric value deduced from the analysis of a source code, and a validation expression defines a logical condition. The quality rule compares the metric with the validation expression:

- If the metric complies with the condition of the validation expression, the quality rule succeeds.
- If the metric does not comply with the condition, the quality rule fails.

ARCAD CodeChecker supplies a number of quality rules via its standard rule sets. All the quality rules included are fully documented. You can create and document your own quality rules from the CodeChecker Studio.

 **Reference**

For more information about metrics, refer to [Metrics on page 69](#).

For more information about validation expressions, refer to [Validation expressions on page 78](#).

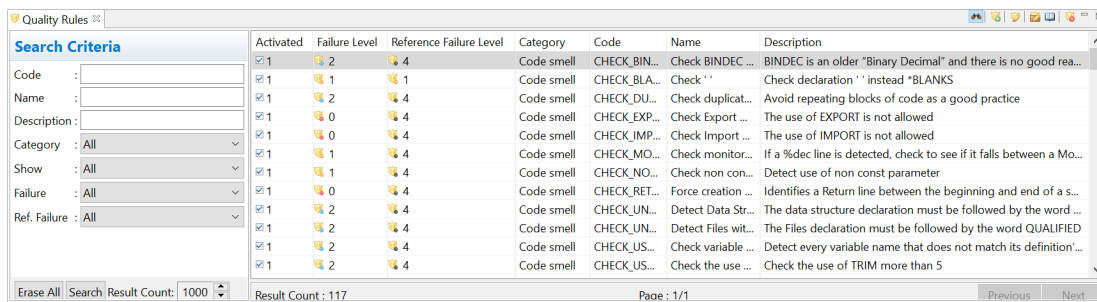




Figure 24: Quality Rules

The  **Quality Rules** view is accessed from the  **Rules Configuration** node in the **Navigator**.

Enter any combination of the above search criteria, then click the **Search** button to display the results. To display the complete list, click the **Search** button without entering any search criteria.


12.1 Creating quality rules


Follow the subsequent steps to create a new quality rule.

Step 1 To access the **Create Quality Rule** wizard, either click the  Create icon in the toolbar of the **Quality Rules** search view, or right-click anywhere in the search list of the view and select  **Create Quality Rule**.


Step 2 Define the quality rule's **Code**, **Name** and **Quality rules**. These values are required to create a new quality rule. The **Code** cannot be changed once the quality rule is created, but the **Name** and **Description** can be edited later.



Click **Finish**.

Result The new quality rule is created and its editor opens automatically. It is displayed in the search list in the  **Quality Rules** search view.


 **Important!**
It is required to edit new quality rules before using them.

12.2 Editing quality rules

To open a quality rule's editor, locate it in the  **Quality Rules** search view, then either:

- double-click on the item in the view,
- right-click on the item in the view and select  **Edit**, or
- select the item in the view and click the  Edit icon in the toolbar.

The editable values are all located in the  **Quality Rule** tab in the quality rule's editor.

The  **Rule Sets** tab displays the list of rule sets that employ the quality rule. Double-click on a rule set to access its editor.

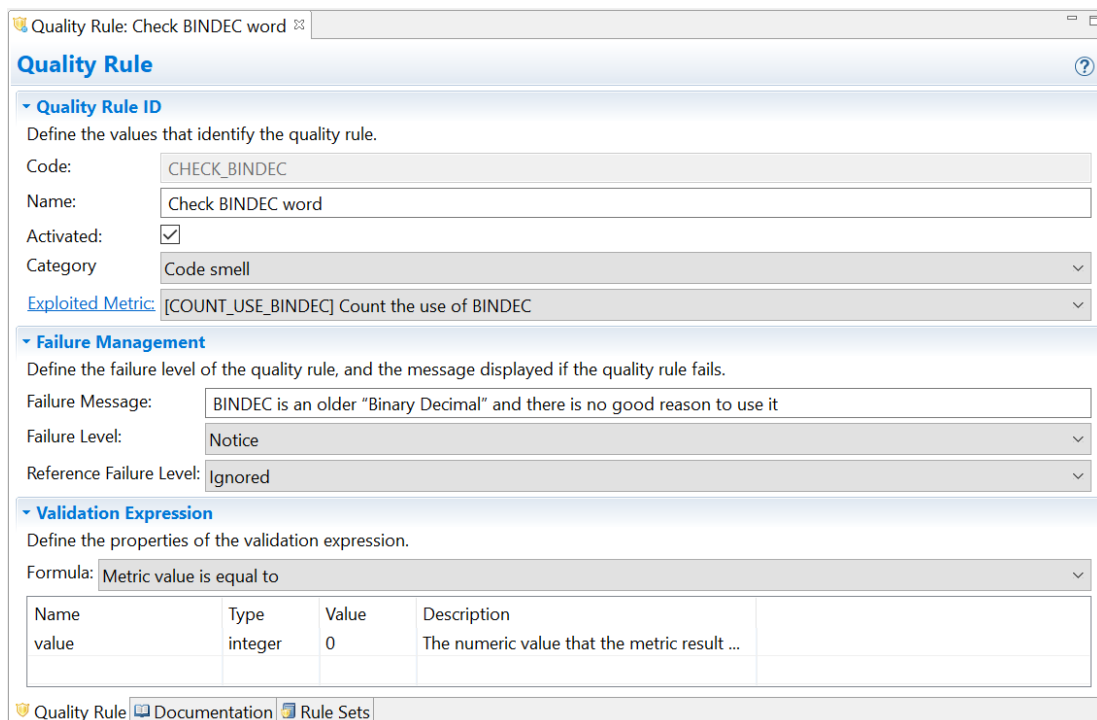


Figure 25: The Quality Rule editor

Quality Rule ID

Code

This code is the quality rule's unique ID and is defined when creating the quality rule. Once a rule is created, it is not possible to edit this code.

Name

The quality rule's name should reflect the condition checked during the code review process to be easily identified.

Activated

When this box is checked, the quality rule is activated. A quality rule must have valid parameters to be activated.

Deactivated quality rules cannot be added to a rule set, and if they are deactivated after being added, they cannot be executed in a campaign.



Warning!

When a quality rule is activated, it cannot be edited anymore.



Reference

For more information about activating quality rules, refer to [Activating quality rules on page 67](#).

Category

Each rule has a corresponding category, that can be changed directly from the CodeChecker Studio. This category value is mainly used when the quality rules are intended to be imported in SonarQube, as they match their rule types.

The four categories available are the following:

- Code smell (maintainability),
- Bug (reliability),
- Vulnerability (security),
- Security hotspot (security).



Reference

For more information about the types of rules in SonarQube, refer to the corresponding [documentation](#).

Exploited Metric

Select the metric to use from the drop-down list. The metric must already exist to be added to a quality rule. You can click the **Exploited Metric** hyperlink to open its editor.

This metric represents a numeric value deduced from the analysis of your source code. The value is compared with the validation expression also defined in the quality rule. If the metric does not comply with the validation expression, the quality rule fails.

**Reference**

For more information about metrics, refer to [Metrics on page 69](#).

Failure Management

Failure Message




This failure message is displayed in the ARCAD CodeChecker plug-in for RDi to provide additional information if the quality rule fails.

The failure message should give information about the logical condition that was not respected and that made the quality rule fail. It should also explain how to fix the issue so the quality rule no longer fails.

Failure Level

Select the failure level of the quality rule from the drop-down list.

There are three hierarchical levels of failure in ARCAD CodeChecker:

-  FATAL, the most critical,
-  WARNING, and
-  NOTICE, the least critical.

The failure level given to a quality rule depends on the code quality that needs to be achieved.

Users with the *Rule Management* role in the CodeChecker Studio determine the failure level to give to each quality rule, and give the appropriate meaning to each failure level.





**Example**

A quality rule can have a NOTICE failure level if this rule should be followed but can be overruled depending on the source code to analyze.

Reference Failure level

Select the reference failure level of the quality rule from the drop-down list.

There are four hierarchical levels of reference failure in ARCAD CodeChecker:

-  FATAL, the most critical,
-  WARNING,
-  NOTICE, the least critical, and
-  IGNORED, the default level.

This reference failure level parameter allows issues found in both the reference and target source codes to be reported with a specific failure level. These issues are usually reported as **Ignored**, along with the failure level set on the rule.

If the Reference failure level parameter is set to any other level than **Ignored**, the issue is then reported using that newly set Reference failure level.

Validation Expression

Formula

Select the validation expression to use from the drop-down list. The expression must already exist to be added to a quality rule.

This validation expression is a logical condition. This expression is compared with the metric, also defined in the quality rule, to return a boolean value (true or false) when the rule is executed. The result of this comparison determines if the quality rule succeeds or fails.

Name, Type & Description

The Name, Type and Description columns give details about which value to give the parameters of the selected formula.


Value



If the validation expression includes specific parameters, enter the Value for each available parameter depending on the logical condition the quality rule should check.



Reference

For more information about validation expressions, refer to [Validation expressions on page 78](#).

12.3 Quality rules batch update

The  **Batch rule update** option allows you to make changes and updates on multiple quality rules at once.

To open the  **Batch update** dialog, select several quality rules in the  **Quality Rules** search view, then either:

- right-click on one of the selected rules and select  **Batch update rules**, or
- click the  Batch update rules icon in the toolbar.

Set each parameter with the values available in the drop-down lists. If you leave one parameter blank, the parameter is not modified for the selected quality rules.

The quality rules parameters available for batch update are the following:

Activated

Sets the activation status of the selected quality rules. The possible values are **Yes** and **No**.

Category

Sets the category of the quality rule, to make sure they correspond to the existing rule types when the quality rules are imported in SonarQube.


Failure level

Sets the failure level of the selected quality rules. The possible values are **Fatal**, **Warning** and **Notice**.

Reference Failure level

Sets the reference failure level of the selected quality rules. The possible values are **Fatal**, **Warning**, **Notice** and **Ignored**.

12.4 Documenting quality rules

Click the  **Documentation** tab in the quality rule's editor to create or modify the rule's documentation. Each rule's documentation should include:

- a description of the rule to summarize what it does and why.
- examples of compliant and non-compliant code.
- any other notes or references to help understand the rule.

The rules' documentation uses the Markdown language. Use `[Ctrl+space]` to open a contextual formatting helper.

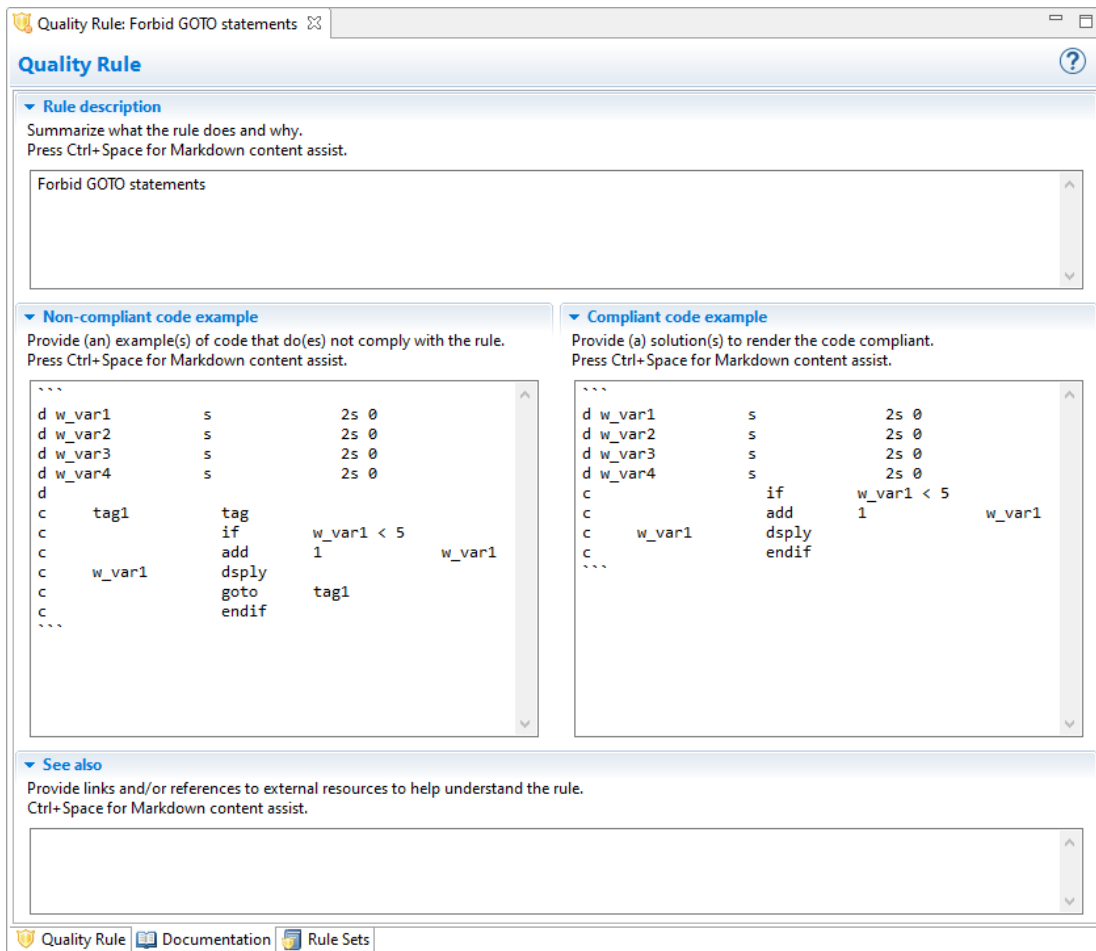




Figure 26: The Quality Rule Documentation editor

You can preview the documentation created here in a web browser. In the Quality rules view, right-click to select a rule and click the  Documentation icon in the contextual menu, or select the rule and click the  Documentation icon in the menu. The rule's documentation is displayed in your default web-browser.

12.5 Activating quality rules

Required role	<input checked="" type="checkbox"/> Activation
---------------	---

Activating a quality rule means this quality rule can be added to rule sets and executed during a code review process. A quality rule must have valid parameters to be activated.

To activate a quality rule, check **Activated** in the quality rule's editor.

When a quality rule is activated, it is no longer possible to edit its parameters, to avoid consistency issues. If a quality rule is activated but needs to be modified, this rule should be deleted and a new one created with the correct parameters.

 **Important!**

Before activating a quality rule, pay attention to the applicable language (s) selected for the exploited metric(s). Active quality rules that can only be applied to specific languages cannot be executed on applications written in another language.

12.6 Adding quality rules to a rule set

Quality rules must be added to rule sets in order to be executed. The quality rules included in a rule set are managed in the rule set's editor.

 **Reference**




For more information about managing included quality rules, refer to [Included Quality Rules on page 58](#).

12.7 Deleting quality rules





 **Warning!**

Deleted quality rules cannot be accessed or recovered.

You cannot delete quality rules that are currently being called by a rule set. You must first remove the rule from all of the sets that employ it before deleting it.

To delete a quality rule, either right-click on it in the  **Quality Rules** search view and select  **Delete**, or select it and click the  Delete icon in the toolbar. Click **OK** to confirm or click **Cancel** to keep the quality rule.

13 Metrics

Required role	 Rule Management
Access	 CodeChecker Server →  Rules Configuration →  Metrics

Chapter Summary

13.1 Creating metrics	70
13.2 Editing metrics	70
13.3 Deleting metrics	72

Metrics are numeric values deduced from the analysis of an application's source code. These values are used in quality rules to help determine code quality.

Metrics are based on metric models. A metric model is a script that defines the actions to carry out and their required properties. The metric created from a metric model must be instantiated to be used in a quality rule: this means values for each of the model's properties are defined to conform to the specific quality rule's context.

The same metric can be used in multiple quality rules, depending on the validation expression defined in each quality rule.

Example

You want to create a quality rule to make sure the word 'TODO' is not found in the source code or the comments. The metric used in the quality rule needs to find the occurrences of this word.

To create the required metric, you will need to use the SEARCH_WORD metric model. When you instantiate the metric, you will define the following properties:

- for **Word**, select **TODO**,
- for **Case sensitive**, select **false**,
- for **Search type**, select **CONTAINS**,
- for **Include comments**, select **true**.

The metric will be incremented by one every time the word 'TODO' is found in a source code or the comments during the execution of the quality rule.

Reference

For more information about metric models, refer to [Metric models on page 73](#).

For more information about quality rules, refer to [Quality rules on page 62](#).

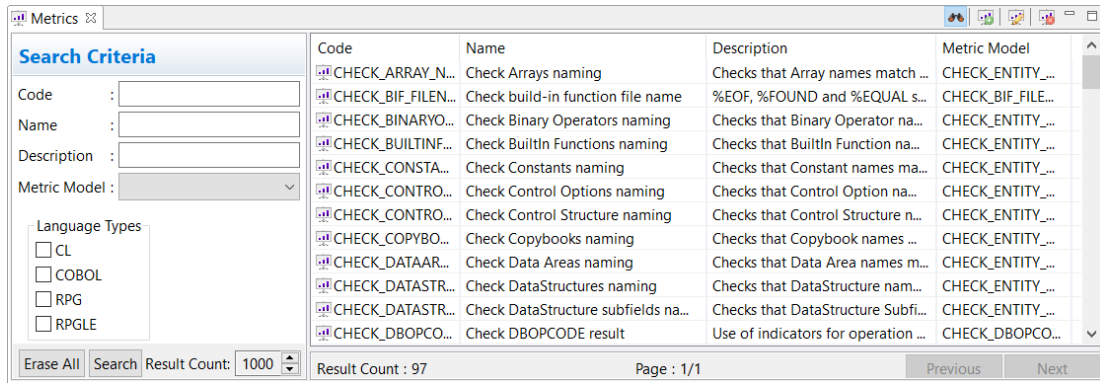





Figure 27: Metrics

The  **Metrics** view is accessed from the  **Rules Configuration** node in the **Navigator**.

Enter any combination of the above search criteria, then click the **Search** button to display the results. To display the complete list, click the **Search** button without entering any search criteria.

13.1 Creating metrics

Follow the subsequent steps to create a new metric.

Step 1 To access the **Create Metric** wizard, either click the  Create icon in the toolbar of the  **Metrics** search view, or right-click anywhere in the search list of the view and select  **Create Metric**.

Step 2 Define the metric's **Code**, **Name** and **Description**. These values are required to create a new metric. The **Code** cannot be changed once the metric is created, but the **Name** and **Description** can be edited later.

Click **Next >** to continue.


Step 3 Select a metric model from the **Available Metric Models** list. This value is required and cannot be edited once the metric is created.

Click **Next >** to continue.

Step 4 Instantiate the metric. These values are required to create a new metric but can be edited later.

- Check the **Languages** the metric applies to.
- Enter a **Value** for each of the metric model properties.

Click **Finish**.


Result The new metric is created and its editor opens automatically. It is displayed in the search list in the  **Metrics** view.



13.2 Editing metrics



Warning!

When a metric is used by one or several *active* quality rules, it cannot be edited anymore. To edit the metric, delete the active quality rule(s) that

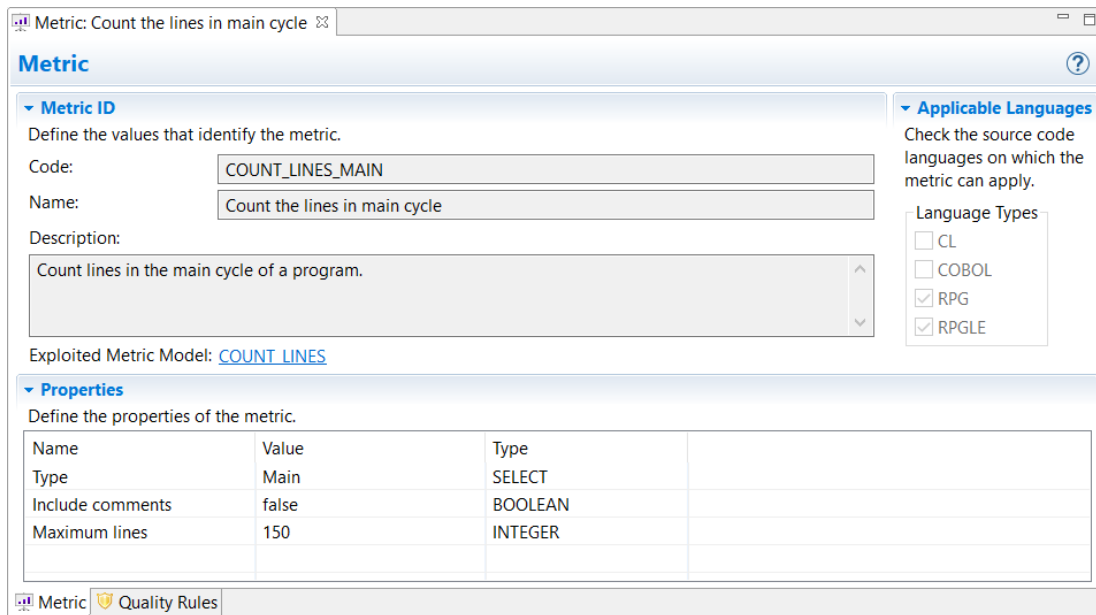
 use it.

To open a metric's editor, locate it in the  **Metrics** search view, then either:

- double-click on the item in the view,
- right-click on the item in the view and select  **Edit**, or
- select the item in the view and click the  Edit icon in the toolbar.

The editable values are all located in the  **Metric** tab in the metric's editor. The  **Quality Rules** tab displays the list of quality rules that employ the metric. Double-click on a quality rule to access its editor.

Save the changes (, Ctrl+S or **File > Save**).



The screenshot shows the 'Metric' editor window for a metric named 'Count the lines in main cycle'. The window has a title bar with the metric name and a search icon. Below the title bar, there are two tabs: 'Metric' (selected) and 'Quality Rules'. The 'Metric' tab is divided into several sections:

- Metric ID:** A section for defining the metric's unique ID. It contains three input fields: 'Code' (with value 'COUNT_LINES_MAIN'), 'Name' (with value 'Count the lines in main cycle'), and 'Description' (with value 'Count lines in the main cycle of a program.'). Below these fields is a link for 'Exploited Metric Model' pointing to 'COUNT_LINES'.
- Applicable Languages:** A section for selecting source code languages. It includes a checkbox for 'Language Types' and a list of languages: 'CL' (unchecked), 'COBOL' (unchecked), 'RPG' (checked), and 'RPGLE' (checked).
- Properties:** A table defining the metric's properties. The table has four columns: 'Name', 'Value', 'Type', and an empty column. The rows are:

Name	Value	Type	
Type	Main	SELECT	
Include comments	false	BOOLEAN	
Maximum lines	150	INTEGER	

Figure 28: The Metric editor

Metric ID

Code

This field displays the metric's unique ID defined when creating the metric. Once a metric is created, it is not possible to edit this code.

Name

The metric's name should reflect the numeric value deduced from the analysis of the source code to be easily identified.

Description

Use this description to give as many details as possible about the numeric value that should be obtained when this metric is used.

Exploited Metric Model

This field displays the metric model selected when creating the metric. To access the metric model's editor, click the metric model's name.

Once a metric is created, it is not possible to edit the metric model selected. If the metric model needs to be changed, the metric should be deleted and a new one created with the correct metric model.

Applicable Languages

Check the language(s) to which the metric applies.

Pay attention to the languages selected and check all the potential languages the metric could apply to. If a certain language is not selected for a metric, the quality rules using this metric will not be executed on applications written in that language.

Example

You create a metric and only check the **RPG** language. You use this metric in a quality rule. This quality rule will not be executed during a code review process if the target application is not written in RPG.

You can track executed rules in the campaign logs. They include messages at verbose level for any skipped rules because they do not apply to your source.

Properties

Name, Type & Description

The **Name**, **Type** and **Description** columns give details on which value to give the properties of the selected metric model.

Value




If the metric model includes specific properties, select a **Value** from the drop-down list or enter it manually for each available property, depending on the numeric value that should be deduced from the source code's analysis.

13.3 Deleting metrics

Warning!

Deleted metrics cannot be accessed or recovered.

You cannot delete metrics that are currently being called by one or more quality rules. You must first delete all of the rules that employ it before deleting the metric.

To delete a metric, either right-click on it in the  **Metrics** search view and select  **Delete**, or select it and click the  Delete icon in the toolbar. Click **OK** to confirm or click **Cancel** to keep the metric.

14 Metric models

Required role	Rule Management
Access	CodeChecker Server → Rules Configuration → Metric Models

Chapter Summary

14.1 Creating metric models	74
14.2 Editing metric models	74
14.3 Deleting metric models	77

Metric models are script instructions used to create metrics and analyze an application's source code. A metric model defines the actions to carry out to retrieve a value when analyzing code. A metric model uses properties. When the metric using a metric model is instantiated for a quality rule, values for each of the properties are defined to conform to the specific rule's context.

Example

You want to create a quality rule to make sure the word 'TODO' is not found in the source code or the comments. The metric used in the quality rule needs to find the occurrences of this word.

To create the required metric, you will need to use the SEARCH_WORD metric model and define the correct properties when you instantiate the metric.

ARCAD CodeChecker comes with a set of standard, ready-to-use, non-editable metric models. However, ARCAD CodeChecker also allows you to create metric models using the Groovy scripting language.

Reference

For more information about the Groovy programming language, refer to the [Groovy documentation](#).

Reference

For more information about metrics, refer to [Metrics on page 69](#).

For more information about quality rules, refer to [Quality rules on page 62](#).

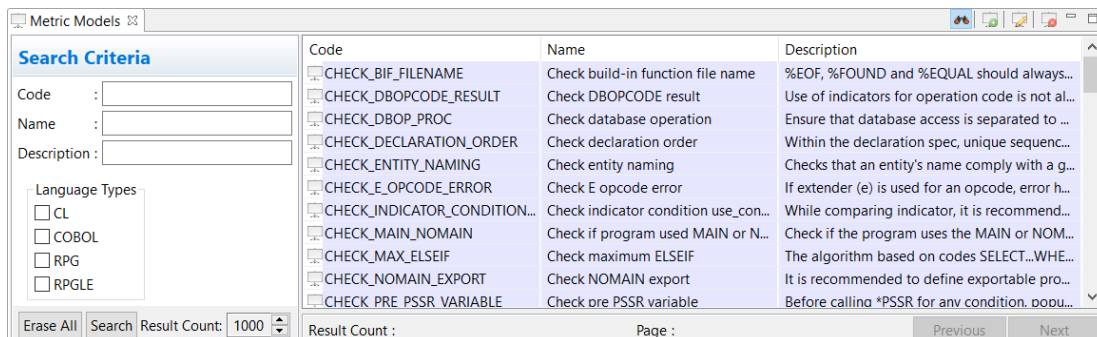


Figure 29: Metric Models




The **Metric Models** view is accessed from the **Configuration** node in the **Navigator**.

The standard metric models supplied by ARCAD CodeChecker are highlighted to be easily identified.

Enter any combination of the above search criteria, then click the **Search** button to display the results. To display the complete list, click the **Search** button without entering any search criteria.


14.1 Creating metric models


Follow the subsequent steps to create a new metric model.

Step 1 To access the **Create Metric Model** wizard, either click the  Create icon in the toolbar of the  **Metric Models** search view, or right-click anywhere in the search list of the view and select  **Create Metric Model**.


Step 2 Define the metric model's **Code**, **Name** and **Description**. These values are required to create new metric models. The **Code** cannot be changed once the metric model is created, but the **Name** and **Description** can be edited later.



Click **Finish**.




Result The new metric model is created and its editor opens automatically. It is displayed in the search list in the  **Metric Models** search view.


 **Important!**
It is required to edit new metric models before using them.

14.2 Editing metric models

To open a metric model's editor, locate it in the  **Metric Models** search view, then either:

- double-click on the item in the view,
- right-click on the item in the view and select  **Edit**, or
- select the item in the view and click the  Edit icon in the toolbar.

You can edit the values located in the  **Metric Model** tab and the  **Script** tab in the metric model's editor. The  **Metrics** tab displays the list of metrics that were created using the metric model. Double click on a metric to access its editor.

 **Warning!**
Any changes made to a metric model are carried over to the metrics already created using this model, *as long as* these metrics are not used in active quality rules. A metric used in an active quality rule can no longer be edited, and any changes made to the metric model used to create it will not be carried over.

Save the changes (, Ctrl+S or **File > Save**).

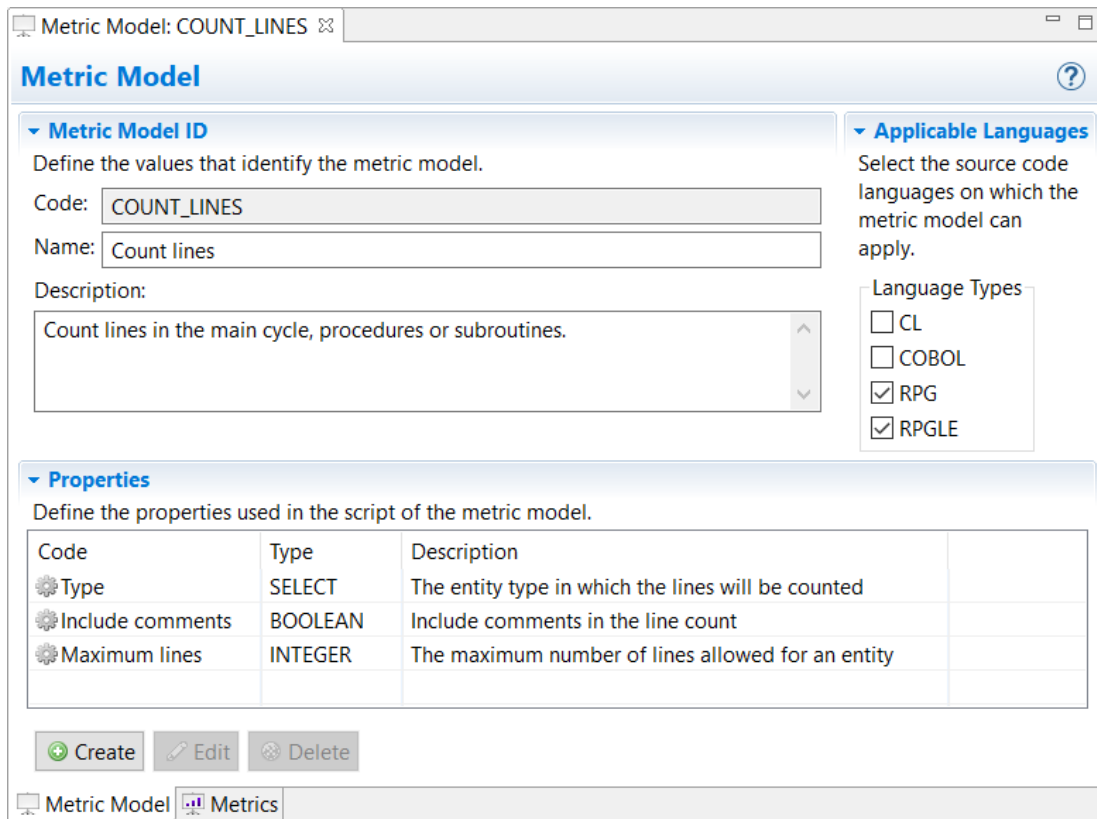


Figure 30: The Metric Model editor

Metric Model ID

Code

This field displays the model's unique ID defined when creating the metric model. Once a model is created, it is not possible to edit this code.

Name

The metric model's name should reflect the actions carried out to retrieve a value during the code analysis to be easily identified.

Description

Use this description to give as many details as possible about the actions carried out by the metric model and the type of value retrieved. This description is displayed in the metric creation wizard.

Applicable Languages

Check the languages to which the metric model applies.

The languages supported by ARCAD CodeChecker are:


- CL
- COBOL
- RPG (RPG III)
- RPGLE (RPG IV)

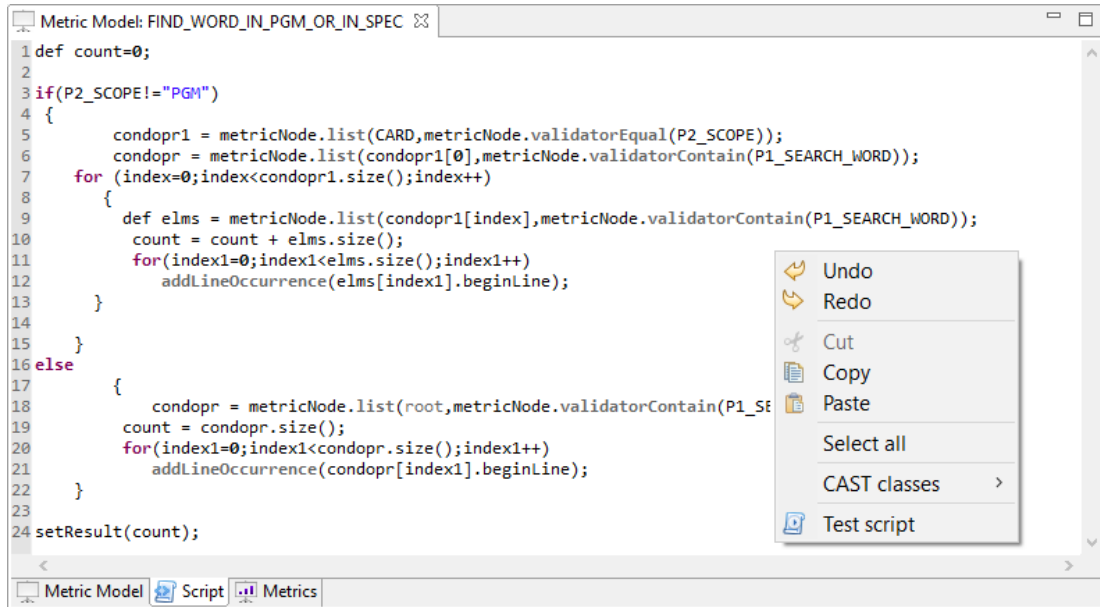
The languages checked will be the only ones available for metrics created using the metric model.

Example

If you leave the **RPGLE** box unchecked for a metric model, you will not be able to apply this language to metrics created using this metric model.

Script

Define the metric model's script. Use the dedicated script editor in the  **Script** tab.



```

1 def count=0;
2
3 if(P2_SCOPE!="PGM")
4 {
5     condopr1 = metricNode.list(CARD,metricNode.validatorEqual(P2_SCOPE));
6     condopr = metricNode.list(condopr1[0],metricNode.validatorContain(P1_SEARCH_WORD));
7     for (index=0;index<condopr1.size();index++)
8     {
9         def elms = metricNode.list(condopr1[index],metricNode.validatorContain(P1_SEARCH_WORD));
10        count = count + elms.size();
11        for(index1=0;index1<elms.size();index1++)
12            addLineOccurrence(elms[index1].beginLine);
13    }
14 }
15 }
16 else
17 {
18     condopr = metricNode.list(root,metricNode.validatorContain(P1_SEARCH_WORD));
19     count = condopr.size();
20     for(index1=0;index1<condopr.size();index1++)
21         addLineOccurrence(condopr[index1].beginLine);
22 }
23
24 setResult(count);
    
```

The screenshot shows a script editor window titled "Metric Model: FIND_WORD_IN_PGM_OR_IN_SPEC". The code is Groovy. A right-click contextual menu is open over the code, showing options: Undo, Redo, Cut, Copy, Paste, Select all, CAST classes (with a right arrow), and Test script.

Figure 31: The Metric Model's Script editor

To insert a class in the script using the **CAST Classes** helper, place the cursor on a line in the script editor. Right-click and select **CAST classes** > *NAME OF THE CLASS* in the contextual menu.

To test the script, right-click anywhere in the script editor and select  **Test script**. A dialog opens to select a source file for testing. Select the file and click **Open**. The test starts and a dialog opens to display the result of the test.

Note

You cannot edit the script of the standard ARCAD CodeChecker metric models.

Reference

For more information about the Groovy programming language, refer to the [Groovy documentation](#).

Properties

Manage the properties included in the metric model's script in the **Properties** section of the editor. The values of these properties are defined in the quality rule(s) that will employ the metrics that use this model.

Follow the subsequent steps to create a new property.

Step 1 To open the **New Metric Model Property** wizard, click  **Create**.

Step 2 Define a **Property ID** for the new property. This ID is used to identify the property in the metric model's script.


Step 3 Select the **Type** of the value from the drop-down list. This value is entered when the metric created from the metric model is instantiated. possible types are:

- Boolean,
- Date,
- Float,
- Integer,
- Select: to define a list of possible choices,
- String.

Step 4 Enter a detailed **Description** of the property. Use this description to give as many details as possible about the value that should be entered when creating a metric, depending on the quality rule's context.

Click **OK**.

Result The new property is created and is displayed in the **Properties** section in the metric model's editor.

To edit a property, either double-click on it, or select it and click  **Edit**. Click **OK** to save any changes made or **Cancel** to keep the property unchanged.

To delete a property, select it and click  **Delete**. Click **Yes** to confirm or **No** to keep the property.

14.3 Deleting metric models




Warning!

Deleted metric models cannot be accessed or recovered.





It is not possible to delete a metric model currently used by a metric. You must first delete all of the metrics that use it before deleting the metric model.

Important!

Standard metric models cannot be deleted.

To delete a metric model, either right-click on it in the  **Metric Models** search view and select  **Delete**, or select it and click the  Delete icon in the toolbar. Click **OK** to confirm or click **Cancel** to keep the metric model.

15 Validation expressions

Required role	 Rule Management
Access	 CodeChecker Server →  Rules Configuration →  Validation Expressions

Chapter Summary

15.1 Creating validation expressions	79
15.2 Editing validation expressions	79
15.3 Deleting validation expressions	81

Validation expressions are logical conditions used in the definition of a quality rule. During the code analysis, the validation expression is compared with the metric also defined in the quality rule to return a boolean value (true or false). The result of this comparison determines if the quality rule succeeds or fails.

ARCAD CodeChecker comes with a set of standard validation expressions that are designed to cover most needs. ARCAD CodeChecker also makes it possible to create specific validation expressions if needed.

Example

You want to create a quality rule to make sure Until statements are not used in a source code.

The validation expression **Metric value is equal to** uses a parameter to set the expected value for a metric. You will use this validation expression in the quality rule, and set the expected value to 0.

After the execution of the quality rule, if the metric value is 0, then the quality rule succeeds. However, if the metric value is different from 0, the quality rule fails because the metric value is not equal to 0, as the validation expression requires.

Reference

For more information about quality rules, refer to [Quality rules on page 62](#).

For more information about metrics, refer to [Metrics on page 69](#).

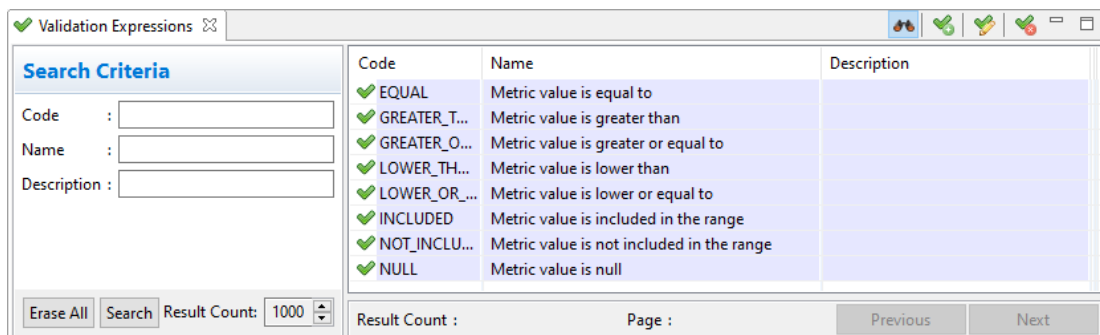




Figure 32: Validation Expressions




The  **Validation Expressions** search view is accessed from the  **Configuration** node in the **Navigator**.

The standard validation expressions are highlighted to be easily identified.

Enter any combination of the above search criteria, then click the **Search** button to display the results. To display the complete list, click the **Search** button without entering any search criteria.


15.1 Creating validation expressions


Follow the subsequent steps to create a new validation expression.

Step 1 To access the **Create Validation Expression** wizard, either click the  Create icon in the toolbar of the  **Validation Expressions** search view, or right-click anywhere in the list of the view and select  **Create Validation Expression**.


Step 2 Define the validation expression's **Code**, **Name** and **Description**. These values are required. The **Code** cannot be changed once the validation expression is created, but the **Name** and **Description** can be edited later.



Click **Finish**.

Result The new validation expression is created and its editor opens automatically. It is displayed in the  **Validation Expressions** search view.

 **Important!**
It is required to edit new validation expressions before using them.

15.2 Editing validation expressions

To open a validation expression's editor, locate the it in the  **Validation Expressions** search view, then either:

- double-click on the item in the view,
- right-click on the item in the view and select  **Edit**, or
- select the item in the view and click the  Edit icon in the toolbar.

Save the changes (, Ctrl+S or **File > Save**).

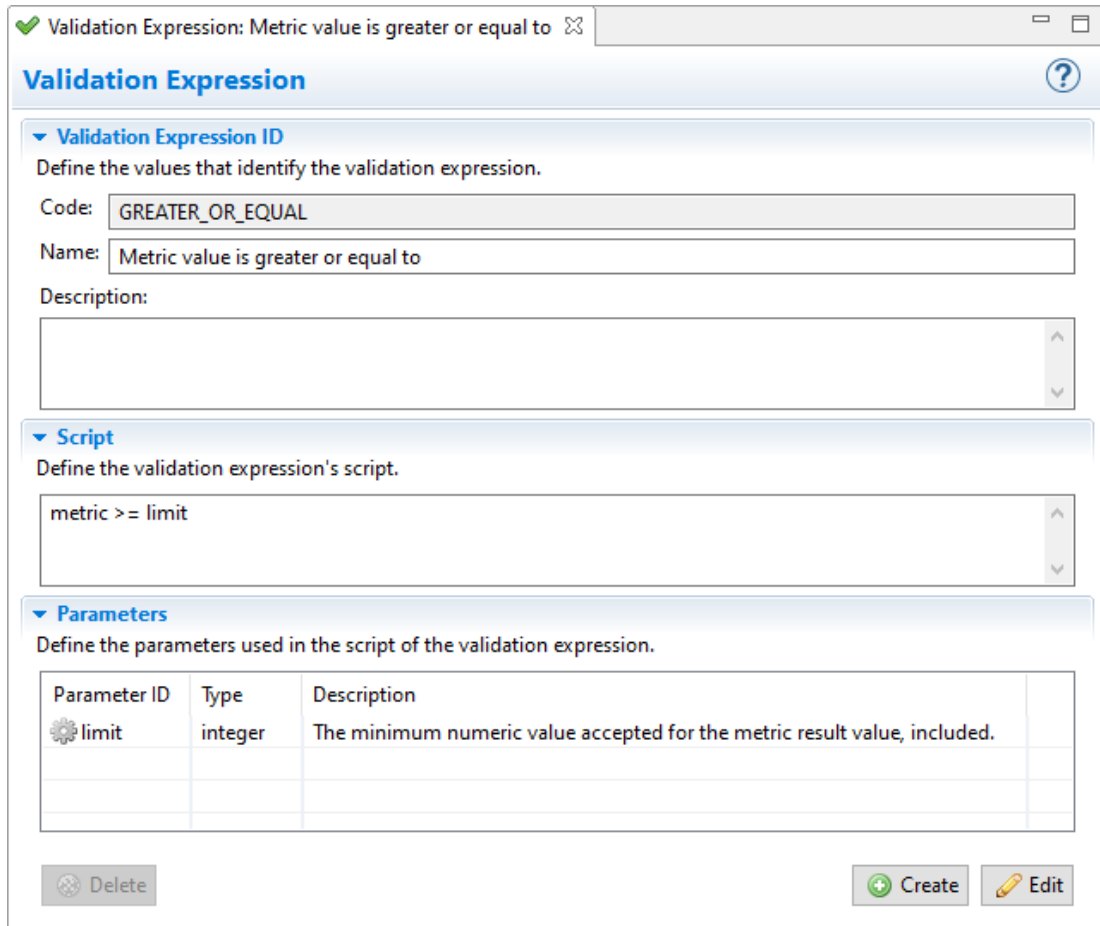


Figure 33: The Validation Expression editor

Validation Expression ID

Code

This field displays the validation expression's unique ID defined when creating the validation expression. Once an expression is created, it is not possible to edit this code.

Name

The validation expression's name should reflect the logical condition defined for the metric comparison to be easily identified.

Description

Use this description to give as many details as possible about the logical condition that should be compared with the metric when this validation expression is used.


Script


Define the logical formula of the validation expression. Use parameters if needed.

Parameters

This section helps you manage the parameters included in the validation expression's script. The values of these parameters are defined in a quality rule.

Follow the subsequent steps to create a new parameter.

- Step 1** To open the **New Validation Expression Parameter** dialog, click  **Create**.
- Step 2** Define a **Parameter ID** for the new parameter. This ID is used to identify the parameter in the validation expression's script.
- Step 3** Select the **Type** of the value from the drop-down list. This value is entered when the quality rules using the validation expression are defined.
- Step 4** Enter a detailed **Description** of the parameter. Use this description to give as many details as possible about the value that should be entered when defining a quality rule, depending on the rule's context.
- Click **OK**.
- Result** The new parameter is created and is displayed in the **Parameters** section in the validation expression's editor.

To edit a parameter, either double-click on it, or select it and click  **Edit**. Click **OK** to save any changes made or **Cancel** to keep the parameter unchanged.

To delete a parameter, select it and click  **Delete**. Click **Yes** to confirm or **No** to keep the parameter.

15.3 Deleting validation expressions




Warning!

Deleted validation expressions cannot be accessed or recovered.

You cannot delete expressions that are currently being called by one or more quality rules. You must first delete all of the rules that employ it before deleting the expression.

Important!

Standard validation expressions cannot be deleted.

To delete a validation expression, either right-click on it in the  **Validation Expressions** search view and select  **Delete**, or select it and click the  Delete icon in the toolbar. Click **OK** to confirm or click **Cancel** to keep the validation expression.



CAMPAIGNS CONFIGURATION

Introduction to code analysis campaigns

A campaign is the association of a code review, by which you define the rules you wish to enforce and the target environment by which you define the code you wish to analyze. ARCAD CodeChecker makes it easy to configure the code review, the target and all the other entities to use for executing code quality campaigns on an application.

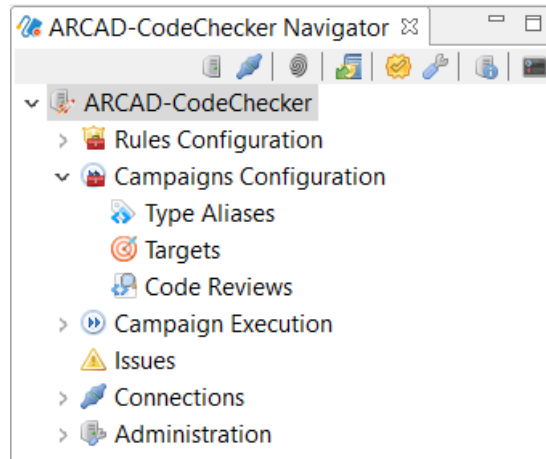






Figure 34: The Campaign Configuration node in the navigator

All of the following entities are accessed and managed in the **Campaign Configuration** node in the navigator, under the currently connected **CodeChecker Server**.

- [Type aliases on page 84](#)
- [Targets on page 91](#)
- [Code reviews on page 86](#)

16 Type aliases

Required role	 Campaign Management
Access	 CodeChecker Server →  Campaign Configuration →  Type Aliases

Chapter Summary

16.1 Creating type aliases	85
16.2 Editing type aliases	85
16.3 Deleting type aliases	85

Custom source types are often found in applications and in order to check the quality of the code for these sources types, ARCAD CodeChecker needs to associate an explicit standard language with them.

Type aliases create the association between a user-defined source type and an actual language-type supported by ARCAD CodeChecker. Once an alias is defined, it allows ARCAD CodeChecker to use the correct parser to analyze source codes with user-defined types.

Warning!
When a source type is unknown, ARCAD CodeChecker ignores the source during analysis.

Example
Your application uses a custom SBRLE source type to define RPGLE subroutines in Copybooks. Type aliases will enable ARCAD CodeChecker to treat SBRLE sources as RPGLE, once the alias SBRLE -> RPGLE is defined.

Reference
For more information about languages, refer to [Presentation of ARCAD CodeChecker on page 15](#).

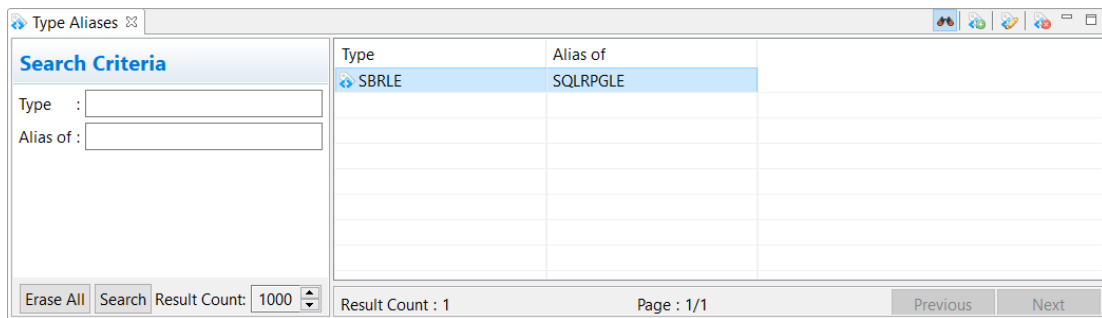






Figure 35: Type aliases

The  **Type Aliases** search view is accessed from the  **Campaign Configuration** node in the **Navigator**.


Enter any combination of the above search criteria, then click the **Search** button to display the results. To display the complete list, click the **Search** button without entering any search criteria.

16.1 Creating type aliases

Follow the subsequent steps to create a new type alias.


Step 1 To access the **Create Type alias** wizard, either click the  Create icon in the toolbar of the **Type Aliases** search view, or right-click anywhere in the list of the view and select  **Create Type alias**.

Step 2 Define the type alias's type. This value is required and cannot be changed once the type alias is created.


 **Important!**
It is not possible to define multiple aliases for the same user-defined type. The database does not allow it.



Step 3 Select the **Type of** language the alias refers to from the drop-down list. This value is required to create a new type alias but can be edited later.

Click **Finish**.

Result The new type alias is created and is displayed in the search list in the  **Type Aliases** view.

16.2 Editing type aliases

To open a type alias's edition dialog, locate it in the  **Type Aliases** search view, then either:

- double-click on the item in the view,
- right-click on the item in the view and select  **Edit**, or
- select the item in the view and click the  Edit icon in the toolbar.

When editing a type alias, only the supported language (**Alias of** field) can be changed. The user-defined **Type** is read-only and cannot be modified.

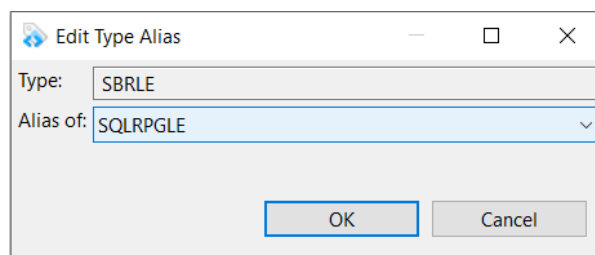










Figure 36: The Type Alias edition dialog

16.3 Deleting type aliases

 **Warning!**
Deleted type aliases cannot be accessed or recovered.

To delete a type alias, either right-click on it in the  **Type Aliases** search view and select  **Delete**, or select it and click the  Delete icon in the toolbar. Click **OK** to confirm or click **Cancel** to keep the type alias.


17 Code reviews

Required role	 Campaign Management
Access	 CodeChecker Server →  Campaign Configuration →  Code Reviews

Chapter Summary


17.1 Creating code reviews	86
17.2 Editing code reviews	87
17.3 Deleting code reviews	90

Code reviews define the ensemble of rule sets that should be executed during a code review process. A code review must be associated with a target to create a campaign. When this campaign is launched, the rule sets defined in the code review are all executed on the associated target.

 **Example**

You want to check the code quality of your application. You have defined quality rules to set code quality goals, and these quality rules are all grouped in the same rule set.

To execute the quality rules on the source code of your application, you need to create a code review to associate the rule set to that target application. You will then be able to generate campaigns using this code review and carry out the code review process.

 **Reference**

For more information about rule sets, refer to [Rule sets on page 55](#).

For more information about targets, refer to [Targets on page 91](#).

For more information about campaigns, refer to [Campaigns on page 105](#).

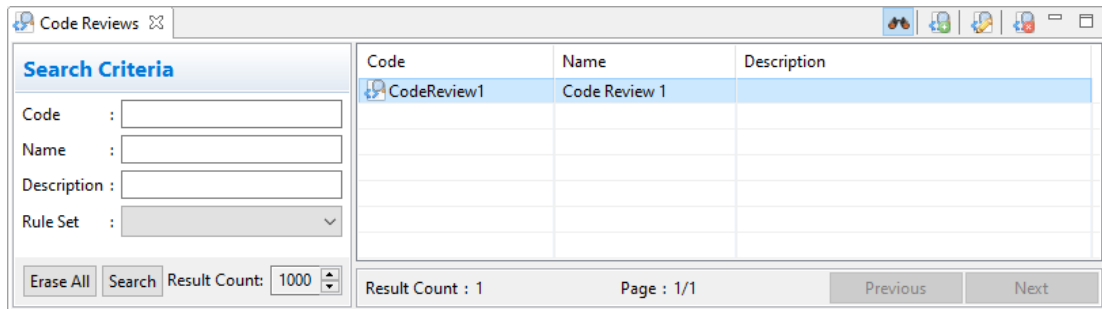





Figure 37: Code Reviews

The  **Code Reviews** view is accessed from the  **Campaign Configuration** node in the **Navigator**. Enter any combination of the above search criteria, then click the **Search** button to display the results. To display the complete list, click the **Search** button without entering any search criteria.


17.1 Creating code reviews


Follow the subsequent steps to create a new code review.

Step 1 To access the **Create Code Review** wizard, either click the  Create icon in the toolbar of the  **Code Reviews** search view, or right-click anywhere in the list and select  **Create Code Review**.

Step 2 Define the code review's **Code** and **Name**. These values are required to create a new code reviews. The **Code** cannot be changed once the code review is created, but the **Name** can be edited later.



Click **Finish**.





Result The new code review is created and its editor opens automatically. It is displayed in the list in the  **Code Reviews** search view.


 **Important!**
It is required to edit the new code review before using it.

17.2 Editing code reviews

To open a code review's editor, locate the code review in the  **Code Reviews** search view, then either:

- double-click on the item in the view,
- right-click on the item in the view and select  **Edit**, or
- select the item in the view and click the  Edit icon in the toolbar.

The editable values are located in the  **Code Review** tab in the code review's editor. The  **Measures** tab allows you to declare the metrics that you wish to follow on the ARCAD CodeChecker dashboard. The  **Schedules** tab displays the list of schedules that use this code review. Double-click on a schedule to access its editor. The  **Issues** tab shows the pending issues linked to this code review.

 **Reference**
For more information about the schedules linked to the code reviews, refer to [Schedules on page 114](#).

For more information about the issues linked to the code reviews, refer to [Issues on page 120](#).

Save the changes (, Ctrl+S or **File > Save**).

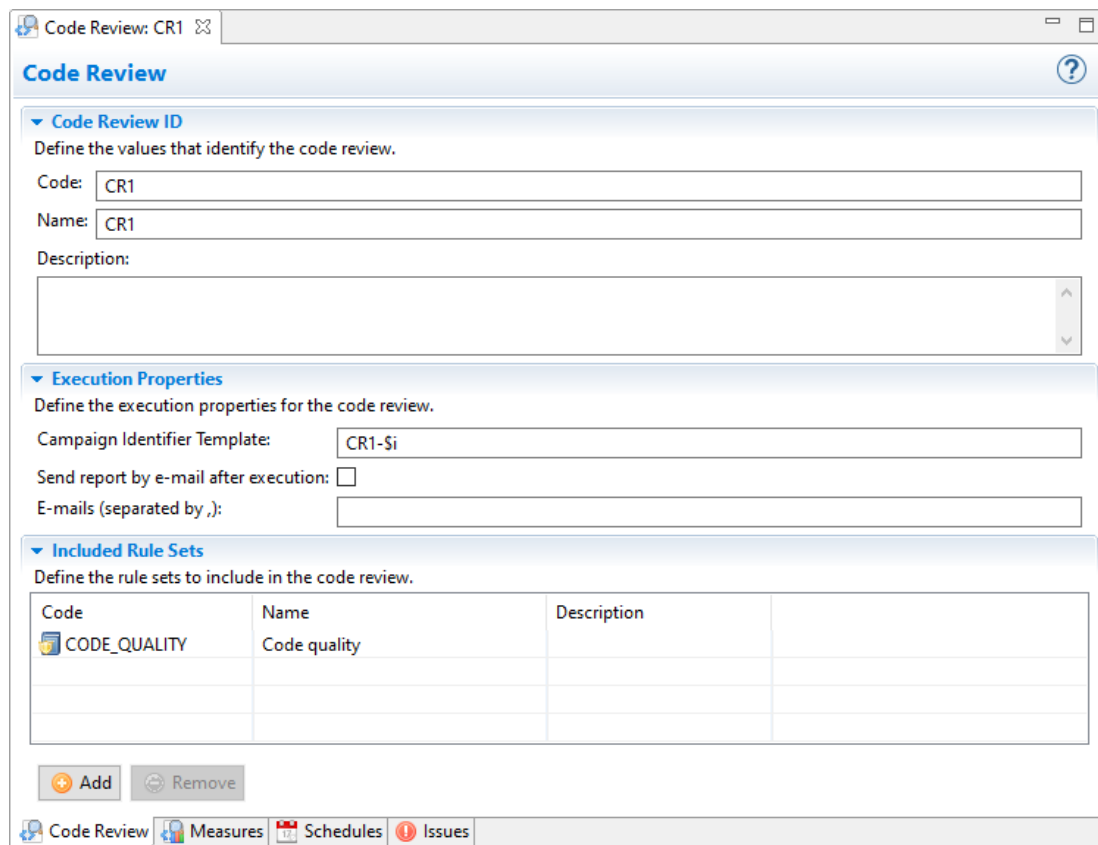


Figure 38: The Code Review editor

17.2.1 Code Review

Code Review ID

Code

This field displays the unique code review's ID defined when creating the code review. Once a code review is created, it is not possible to edit this code.

This code can be used in the ARCAD CodeChecker plug-ins for Jenkins and for SonarQube.

Name

The code review's name should reflect the context of the code quality analysis to be easily identified

Description

Use this description to give as many details as possible about the ensemble of rule sets included in the code review

Execution Properties

Campaign Identifier Template

Define the default identifier to assign to every campaign that is created using the code review. This code will be the campaign's ID.

Use the variable `$(i)` in this field to track the number of campaigns created using the code review. The variable is incremented by one every time a new campaign is created.

Send Recap After Execution

When this box is checked, a recap is sent by email every time a campaign that uses this code review is launched. This recap is sent to the addresses listed in the **Emails** field. Add as many email addresses as needed in the **Emails** field, separated with semicolons.

This recap email contains the same information as the results displayed in a campaign's editor.



Reference

For more information about accessing and understanding campaign results, refer to [Understanding campaign results on page 108](#).


Included Rule Sets

This section enables you to manage the rule sets included in the code review.

To add a rule set to the code review, click  **Add**. In the  **Rule Sets** selection dialog, select one or several sets [Ctrl+click] then click **OK**.



Important!

Only activated rule sets can be added to a code review. The rule sets that are not activated will not be displayed in the  **Rule Sets** selection dialog.



Reference

For more information about activating rule sets, refer to [Activating rule sets on page 58](#).

To remove a rule set from the code review, select the item and click  **Remove**.

17.2.2 Measures

This section enables you to declare the metrics that you wish to follow on ARCAD Dashboard.

The list of metrics defined in the  **Code Reviews** editor is evaluated during a campaign execution and the history of the results are kept between each campaign execution. When measures are calculated, the results are kept in relation to the target defined for the campaign.




Example

You run a modernization set of quality rules on your source code, but you still wish to monitor some general aspects of your development like the number of lines or the number of comments in your code.

Define the metrics you wish to monitor in the code review and display the results in a dashboard.

Note

The metrics of the quality rules added to the code review are also measured for ARCAD Dashboard.

To add a metric to the measure list, click  **Add**. In the **Metrics** selection dialog, select one or several metrics *[Ctrl+click]* then click **OK**.

To remove a metric from the measure list, select the item and click  **Remove**.




Reference

For more information about ARCAD Dashboard, refer to the [documentation](#).





17.3 Deleting code reviews

Warning!

Deleted code reviews cannot be recovered. It is not possible to delete a code review if it is used in one or several campaigns and/or schedules.

To delete a code review, either right-click on it in the  **Code Reviews** search view and select  **Delete**, or select it and click the  Delete icon in the toolbar. Click **OK** to confirm or click **Cancel** to keep the code review.

18 Targets

Required role	 Campaign Management
Access	 CodeChecker Server →  Campaign Configuration →  Targets

Chapter Summary

18.1 Creating targets	92
18.2 Editing targets	92
18.3 Resetting targets	102
18.4 Deleting targets	102

Targets identify the applications and the specific source members to analyze during a code review process. A target must be associated with a code review to create a campaign. When this campaign is launched, the rule sets defined in the code review are all executed on the associated target.

There are different types of targets:

1. IBM i targets, which connect to IBM i and specify the libraries, objects and members to include in the code review process.
2. ARCAD targets, which connect to an ARCAD server and specify the application, environment and version to include in the code review process.
3. Git targets, which connect to a Git repository, clone it to include its content in the code review process.
4. External targets, that are generated when using Jenkins to upload source and run code review campaigns.

Reference

For more information about code reviews, refer to [Code reviews on page 86](#).

For more information about campaigns, refer to [Campaigns on page 105](#).

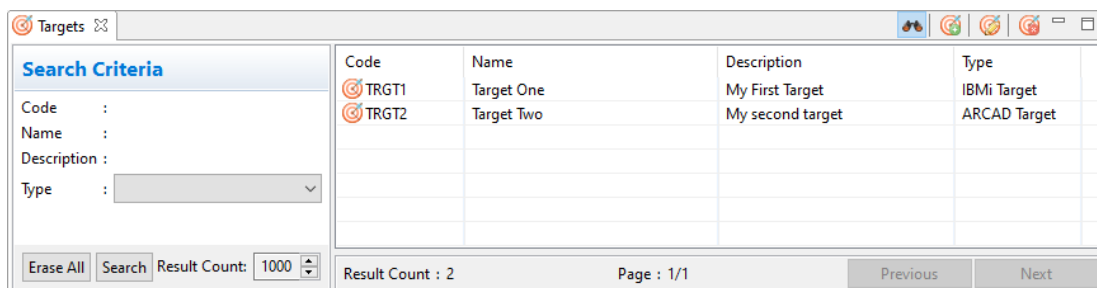





Figure 39: Targets

The  **Targets** view is accessed from the  **Campaign Configuration** node in the **Navigator**.

Enter any combination of the above search criteria, then click the **Search** button to display the results. To display the complete list, click the **Search** button without entering any search criteria.

18.1 Creating targets

Follow the subsequent steps to create a new target.

Step 1 To access the **Create Target** wizard, either click the  Create icon in the toolbar of the  **Targets** search view, or right-click anywhere in the list of the view and select  **Create Target**.

Step 2 Define the target's **Code** and **Name**. These values are required to create a new target but can be edited later.


Step 3 Enter a **Description** for the target.

Step 4 Select the target's **Type** from the drop-down list. This value is required and cannot be edited once the target is created.

- Select **IBM i Target** if the target is an IBM i program.
- Select **ARCAD Target** if the target is an ARCAD application.
- Select **Git Target** if the target is a Git repository.
- Select **External Target** if the sources and execution will be managed by Jenkins.

Step 5 Tick the **Upload to SonarQube** box if your code quality results are managed via SonarQube. This option is enabled only if the [ARCAD CodeChecker for SonarQube](#) solution is set up.


Click **Finish**.



Result The new target is created and its editor opens automatically. It is displayed in the list in the  **Targets** search view.



 **Important!**

It is required to edit new targets before using them.

18.2 Editing targets

To open a target's editor, locate the target in the  **Targets** search view, then either:

- double-click on the item in the view,
- right-click on the item in the view and select  **Edit**, or
- select the item in the view and click the  Edit icon in the toolbar.

The editable values are all located in the  **Target** tab in the target's editor. The  **Schedules** tab displays the list of schedules that are using this target. Double-click on a schedule to access its editor.

 **Reference**

For more information about the schedules linked to the targets, refer to [Schedules on page 114](#).

Save the changes (, Ctrl+S or **File > Save**).

There are four types of targets, and each type requires different parameters for the target to be used.

1. [IBM i targets](#)
2. [ARCAD targets](#)

3. [Git targets](#)
4. [External targets on page 101](#)

18.2.1 IBM i targets

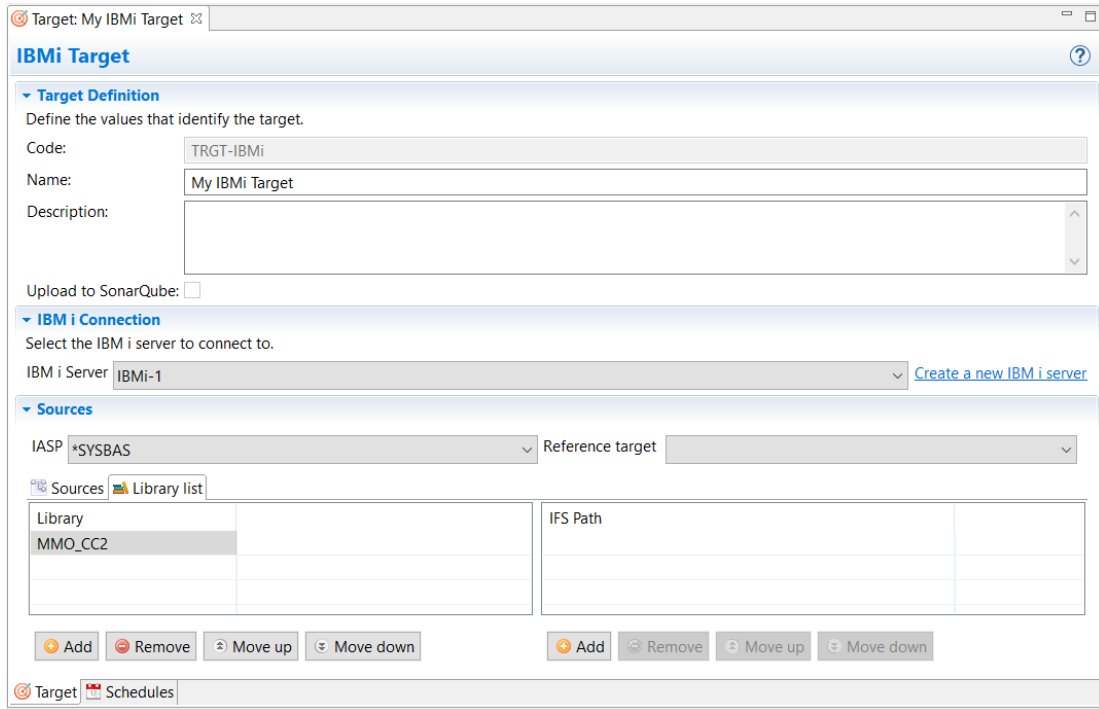


Figure 40: The Target editor - IBM i targets

Target Definition

Name

The target's name should reflect the application to analyze during a code review process to be easily identified.

Description

Use this description to give as many details as possible about the application and the source members that will be analyzed during a code review process.

Upload to SonarQube

If ticked, the issues raised for this target are managed via SonarQube. This option is only available if the [ARCAD CodeChecker for SonarQube](#) solution is set up.

IBM i Connection

Select the IBM i server from the drop-down list. If the connection does not exist, click the link to **Create a new IBM i server**.

Reference

IBM i connections are managed from the IBM i Servers view. For more information, refer to [IBM i servers on page 47](#).

Reference target

Declare a **Reference target** to manage the delta between issues raises when executing a campaign.

A campaign for an IBM i target with a reference will be comprised of two executions:

- One execution to list the issues from the source members from the reference (if it exists). These issues will be marked as **Ignored** by default.
- One execution to list the issues from the target's library members. Issues already found in the reference will still be **Ignored**, and new issues (ie. not found in the reference) will be marked as **Opened**.

The members from the target will simply be looked up in the reference based, by looking for the couple file/member in the reference target's libraries.

Note

Leave the reference target blank if you do not want to compute delta for the target. The campaign execution will act as before, counting every issue as an **Open** issue.

Note

When a target is used as a reference, it cannot define a reference for itself (the selection drop-down list will be disabled). Also, a target using a reference cannot be used as a reference (it will not appear in the drop-down list of reference target).

Sources

Specify the sources to include in the code review process. Follow the subsequent steps to add sources to a target.

Step 1 Select the **IASP** (independent axillary storage pool) from the drop-down list, if the IBM i objects linked to this project are located on an iASP. The list shows all the available IASP on the selected IBM i server. The list is refreshed when the server selection is changed.

Step 2 In the **Sources** tab, click the  Add button.

Step 3 Select the source path type in the dialog.

Tick the **Library** button to add sources from a library.

Tick the **IFS path** button to add sources from the integrated file system.

Click **OK**.

Step 4 Define the source path according to its type:

Library: Define the pattern for the libraries, files, and members you wish to add.

IFS Path: Define an IFS pattern path, using wildcards in any part of the path to make multiple selections.

 **Important!**

Each definition is a "Path pattern", whether it is a Library path or an IFS path. Each pattern supports two types of wildcards:

- *******: one or many character of any type
- **?**: one character of any type

A blank field is equivalent to *******, which means "everything".

Click **OK**.

Result The source is added to the target. It is displayed in the **Sources** tab in the target's editor.

To delete sources from a target, select the path in the list and click the  **Remove** button.

Library list

Use the **Library list** tab to indicate where **/COPY** and **/INCLUDE** sources can be found.

Library list

List the additional libraries in which to search for the sources called by the **/COPY** and **/INCLUDE**.

IFS path

Specify the path in the IFS (Integrated File System) where the sources called by the **/COPY** and **/INCLUDE** are.

Reorder the lists as the sources will be processed in the order in the list.

18.2.2 ARCAD targets

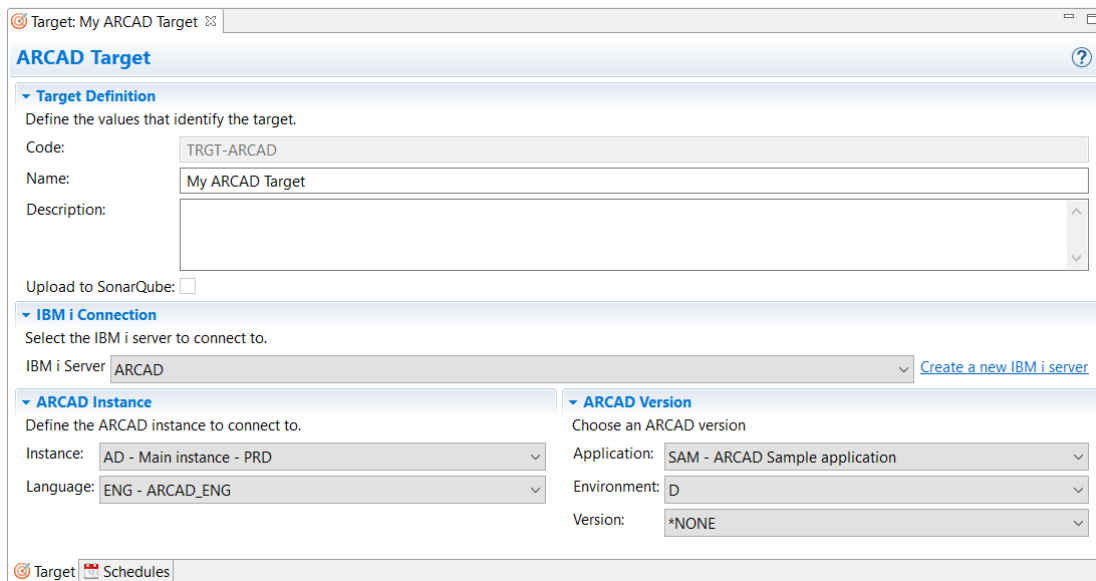


Figure 41: The Target editor - ARCAD targets

Target Definition

Name

The target's name should reflect the application to analyze during a code review process to be easily identified.

Description

Use this description to give as many details as possible about the application and the source members that will be analyzed during a code review process.

Upload to SonarQube

If ticked, the issues raised for this target are managed via SonarQube. This option is only available if the [ARCAD CodeChecker for SonarQube](#) solution is set up.

IBM i Connection

Select the IBM i server from the drop-down list. If the connection does not exist, click the link to **Create a new IBM i server**.

Reference

IBM i connections are managed from the IBM i Servers view. For more information, refer to [IBM i servers on page 47](#).

ARCAD Instance

Instance

Select the name of the ARCAD instance from the drop-down list.

Language

Enter the language defined for the ARCAD instance manually, or select it from the drop-down list.

ARCAD Version

Application

Enter the code of the ARCAD application manually, or select it from the drop-down list.

Environment

Enter the code of the environment where the ARCAD application is located manually, or select it from the drop-down list. Set this option to ***NONE** to define the ARCAD target for an entire environment.

Version

Enter the version number of the ARCAD application manually, or select it from the drop-down list. Set this option to ***NONE** to define the ARCAD target for an entire environment.

18.2.3 Git targets

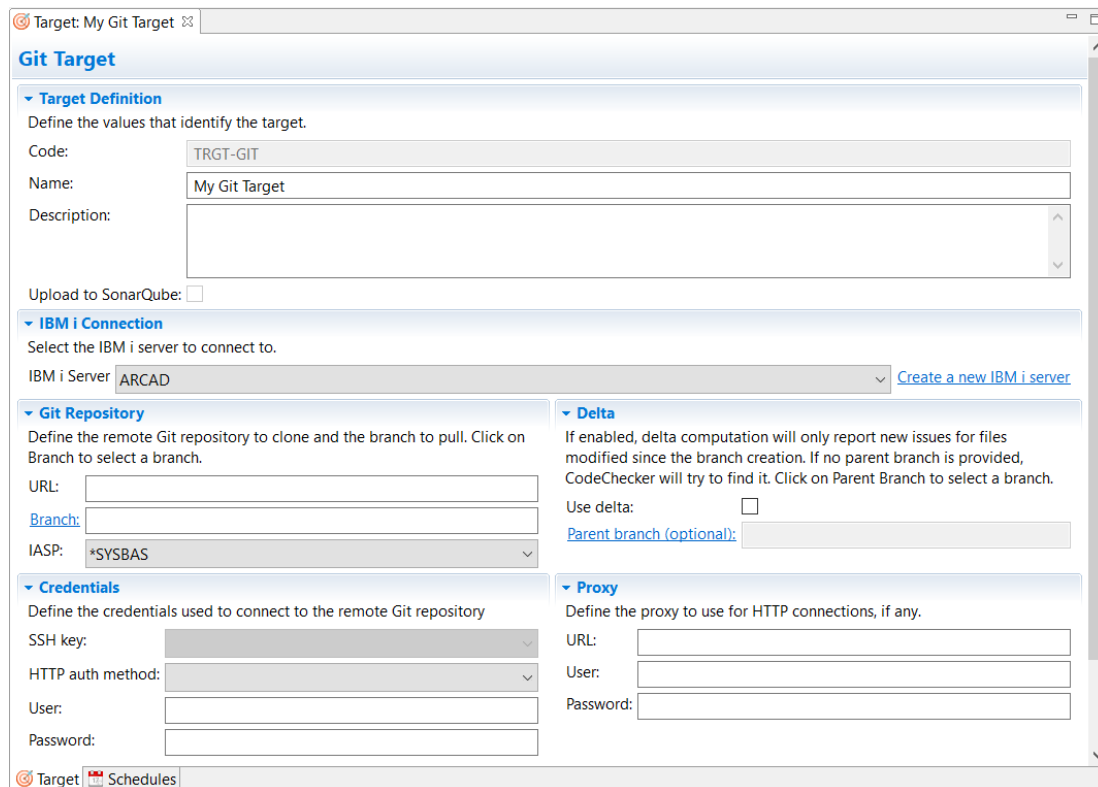


Figure 42: The Target editor - Git targets

Target Definition

Name

The target's name should reflect the application to analyze during a code review process to be easily identified.

Description

Use this description to give as many details as possible about the application and the source members that will be analyzed during a code review process.

Upload to SonarQube

If ticked, the issues raised for this target are managed via SonarQube. This option is only available if the [ARCAD CodeChecker for SonarQube](#) solution is set up.

IBM i Connection

Select the IBM i server from the drop-down list. If the connection does not exist, click the link to **Create a new IBM i server**.

 **Reference**

IBM i connections are managed from the IBM i Servers view. For more information, refer to [IBM i servers on page 47](#).

Git Repository

Define the remote Git repository to clone and the branch to pull. Click on a branch to select it.

URL

Enter URL to the Git repository. Depending on the protocol used to clone the repository (HTTP or SSH), the other fields of the editor will be enabled or disabled.

Branch

Enter the name of the branch to clone or click the **Branch** link to select the branch from a dialog.

IASP

Select the **IASP** (independent axillary storage pool) from the drop-down list, if the IBM i objects linked to this project are located on an IASP. The list shows all the available IASP on the selected IBM i server. The list is refreshed when the server selection is changed.

Credentials

Depending on the connection protocol you choose (SSH or HTTP), define the credentials used to connect to the remote git repository.

SSH key

Select the SSH key in the drop-down list. The SSH keys available are the one configured in the [SSH Keys](#) view.

HTTP auth method

When connecting through HTTP, an authentication method must be selected. Available methods are:

- **Basic auth:** requires a user name and a password
- **Personal token:** requires a user name; the token must be put in the password field.
- **OAuth:** the OAuth token must be put in the connection URL as well as in the password field. (Example URL: https://my_token@github.com/user/repo.git).
- **SAML:** the access token must be put in the password field. It will be used as a user credential.
- **SAML (header):** the access token must be put in the password field. It will be put in the HTTP header with each request.

User & Password

Enter the user and password to connect to the Git repository.

Proxy

This section is only enabled when connecting through HTTP. Define the proxy server's URL and the credentials to use to connect to it.

Delta

There are two possible ways to use a Git target in a campaign: full mode or delta mode.

Full mode: When the Use Delta option is disabled on a target, the remote branch is checked out entirely and all the source files are analyzed during a campaign.

Delta mode: When the Use Delta option is enabled on a target, only the files that have been modified on that branch are analyzed. First, the repository is checked out and only the modified files are analyzed. Then, the repository is checked out again using the initial commit id (i.e. the root of the branch) and the modified files get analyzed again. Issues found in both the tip and the root of the branch are ignored by default.

Tick the **Use delta** box to enable the delta mode. You can set the name of the **Parent Branch** or click its hyperlink to open a selection dialog to pick an existing branch. If no parent branch is provided, ARCAD CodeChecker tries to find it.

The **Initial Commit ID** field is read only and filled after the target has been used in a campaign at least once. It contains the ID of the commit detected by ARCAD CodeChecker as the commit on which the branch was created.

18.2.4 External targets

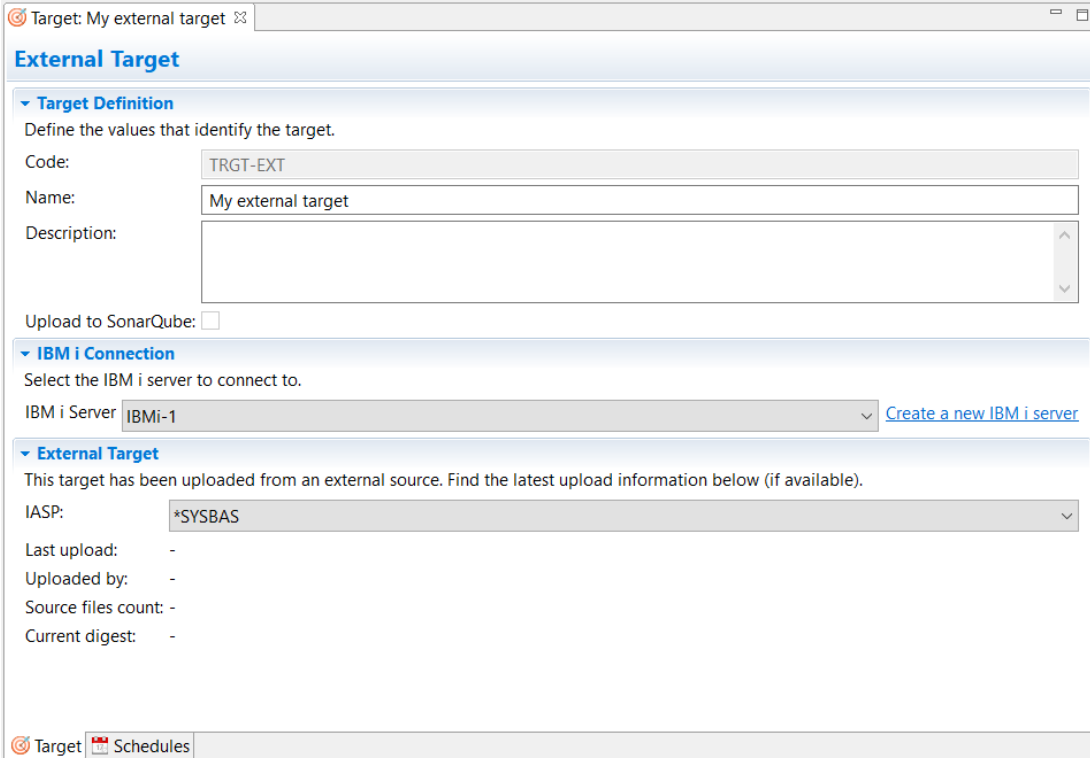


Figure 43: The Target editor - External targets

Target Definition

Name

The target's name should reflect the application to analyze during a code review process to be easily identified.

Description

Use this description to give as many details as possible about the application and the source members that will be analyzed during a code review process.

Upload to SonarQube

If ticked, the issues raised for this target are managed via SonarQube. This option is only available if the [ARCAD CodeChecker for SonarQube](#) solution is set up.

IBM i Connection


Select the IBM i server from the drop-down list. If the connection does not exist, click the link to **Create a new IBM i server**.

Reference

IBM i connections are managed from the IBM i Servers view. For more information, refer to [IBM i servers on page 47](#).





External target are managed outside of the CodeChecker Studio. Information about the target are uploaded from the external source.

18.3 Resetting targets

The  **Reset Target** option makes it possible to delete all the issues and measures linked to the target. You can reset several targets at once.

Warning!

Deleted related issues and measures cannot be recovered.

To reset a target, either right-click on the target in the  **Targets** search view and click  **Reset Target**, or select the target in the  **Targets** search view and click the  Reset Target icon in the toolbar.

A confirmation dialog then pops up, click **Yes** to confirm or click **No** to cancel.

18.4 Deleting targets

Warning!


Deleted targets cannot be recovered.

To delete a target, either right-click on it in the  **Targets** search view and select  **Delete**, or select it and click the  Delete icon in the toolbar. Click **OK** to confirm or click **Cancel** to keep the target.



EXECUTING CAMPAIGNS

Introduction to code quality campaigns executions

All of the entities created and configured in the  **Configuration** node define how to check the code quality of an application. In order to carry out the code review process, the combination of rule sets containing the desired quality rules must be defined, as well as the target application that needs to be analyzed. These two entities can then be associated in a campaign, whether the campaign is created manually or scheduled. When the campaign is launched, the rule sets will be executed on the associated target.

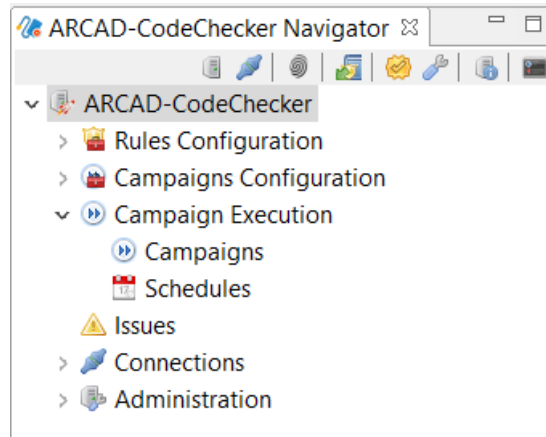








Figure 44: The Campaign Execution node in the navigator

All of the following entities are accessed and managed in the  **Campaign Execution** node in the navigator, under the currently connected  **CodeChecker Server**.

-  [Campaigns on page 105](#)
-  [Schedules on page 114](#)
-  [Issues on page 120](#)

19 Campaigns

Required role	 Campaign Execution
Access	 CodeChecker Server →  Campaign Execution →  Campaigns

Chapter Summary

19.1 Creating campaigns manually.....	106
19.2 Duplicating campaigns.....	107
19.3 Launching campaigns manually.....	107
19.4 Understanding campaign results.....	108
19.5 Exporting campaign results.....	112
19.6 Emailing campaign results.....	113
19.7 Deleting campaigns.....	113

Campaigns are used to carry out a code review process and to keep records of these executions over time. Associate a code review with a target to define the context for each campaign's analysis. When a campaign is launched, the rule sets defined in the selected code review are executed on the associated target to check the code quality.

Once a campaign is completed, the campaign status indicates if the quality rules contained in the executed rule sets succeeded or failed. The campaign results can then be viewed to know which quality rules failed and which source members are concerned.

Example

You want to check the code quality of your application. You have defined quality rules to set code quality goals, and these quality rules are grouped into a rule set.

To execute the quality rules on the source code of your application, you need to create a code review to associate the rule set to that target application. You will then be able to generate campaigns using this context and carry out the code review process.

Reference

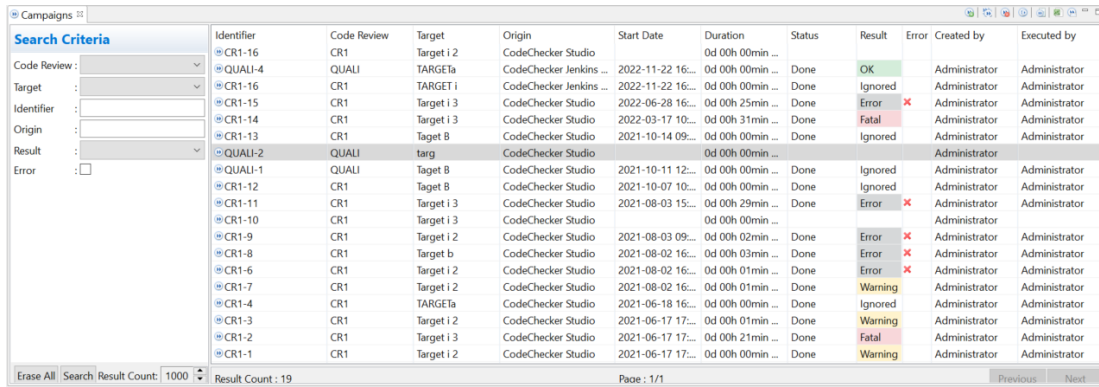
For more information about code reviews, refer to [Code reviews on page 86](#).

For more information about targets, refer to [Targets on page 91](#).

Campaigns can be manually created and launched, or scheduled to automate the creation and launch processes.

Reference

This chapter describes how to manually create and launch campaigns. For more information about scheduling campaigns, refer to [Schedules on page 114](#).



Search Criteria	Identifier	Code Review	Target	Origin	Start Date	Duration	Status	Result	Error	Created by	Executed by
Code Review :	CR1-16	CR1	Target i 2	CodeChecker Studio	2022-11-22 16...	0d 00h 00min ...	Done	OK		Administrator	Administrator
Target :	QUALI-4	QUALI	TARGETa	CodeChecker Jenkins ...	2022-11-22 16...	0d 00h 00min ...	Done	Ignored		Administrator	Administrator
Identifier :	CR1-16	CR1	TARGET i	CodeChecker Studio	2022-11-22 16...	0d 00h 00min ...	Done	Ignored		Administrator	Administrator
Origin :	CR1-15	CR1	Target i 3	CodeChecker Studio	2022-06-28 16...	0d 00h 25min ...	Done	Error	✘	Administrator	Administrator
Result :	CR1-14	CR1	Target i 3	CodeChecker Studio	2022-03-17 10...	0d 00h 31min ...	Done	Fatal		Administrator	Administrator
Error :	CR1-13	CR1	Target B	CodeChecker Studio	2021-10-14 09...	0d 00h 00min ...	Done	Ignored		Administrator	Administrator
	QUALI-2	QUALI	tarig	CodeChecker Studio	2021-10-11 12...	0d 00h 00min ...	Done	Ignored		Administrator	Administrator
	QUALI-1	QUALI	Target B	CodeChecker Studio	2021-10-07 10...	0d 00h 00min ...	Done	Ignored		Administrator	Administrator
	CR1-12	CR1	Target B	CodeChecker Studio	2021-10-07 10...	0d 00h 00min ...	Done	Ignored		Administrator	Administrator
	CR1-11	CR1	Target i 3	CodeChecker Studio	2021-08-03 15...	0d 00h 29min ...	Done	Error	✘	Administrator	Administrator
	CR1-10	CR1	Target i 3	CodeChecker Studio	2021-08-03 09...	0d 00h 00min ...	Done	Error	✘	Administrator	Administrator
	CR1-9	CR1	Target i 2	CodeChecker Studio	2021-08-02 16...	0d 00h 03min ...	Done	Error	✘	Administrator	Administrator
	CR1-8	CR1	Target b	CodeChecker Studio	2021-08-02 16...	0d 00h 01min ...	Done	Error	✘	Administrator	Administrator
	CR1-6	CR1	Target i 2	CodeChecker Studio	2021-08-02 16...	0d 00h 01min ...	Done	Warning		Administrator	Administrator
	CR1-7	CR1	Target i 2	CodeChecker Studio	2021-08-02 16...	0d 00h 01min ...	Done	Warning		Administrator	Administrator
	CR1-4	CR1	TARGETa	CodeChecker Studio	2021-06-18 16...	0d 00h 00min ...	Done	Ignored		Administrator	Administrator
	CR1-3	CR1	Target i 2	CodeChecker Studio	2021-06-17 17...	0d 00h 01min ...	Done	Warning		Administrator	Administrator
	CR1-2	CR1	Target i 3	CodeChecker Studio	2021-06-17 17...	0d 00h 21min ...	Done	Fatal		Administrator	Administrator
	CR1-1	CR1	Target i 2	CodeChecker Studio	2021-06-17 17...	0d 00h 00min ...	Done	Warning		Administrator	Administrator

Figure 45: Campaigns

The  **Campaigns** view is accessed from the  **Campaign Execution** node in the **Navigator**.

Enter any combination of the above search criteria, then click the **Search** button to display the results. To display the complete list, click the **Search** button without entering any search criteria.

19.1 Creating campaigns manually



Follow the subsequent steps to manually create a new campaign.

Note

When a campaign is created and correctly configured, it is also possible to duplicate it to quickly create predefined entities.

Reference

For more information about duplicating campaigns, refer to [Duplicating campaigns on the facing page](#).

Step 1 To access the **Create Campaign** wizard, either click the  **Create** icon in the toolbar of the **Campaigns** search view, or right-click anywhere in the search list of the view and select  **Create Campaign**.

Step 2 Select a **Code Review** from the drop-down list. This value is required and cannot be edited once the campaign is created.

When the campaign is launched, the rule sets defined in the selected code review will be executed on the associated target.


Step 3 Define the campaign's **Identifier**. This value is required and cannot be edited once the campaign is created.

By default, the code value is automatically defined according to the [Campaign Identifier Template](#) set for the selected code review.

Step 4 Select a **Target** from the drop-down list. This value is required and cannot be edited once the campaign is created.

When the campaign is launched, the code review will be carried out on the selected target application.

Click **Finish**.


Result The new campaign is created, is displayed in the list in the  **Campaigns** search view, and is ready to be launched.

 **Note**

Before a campaign is launched, its **Start Date**, **Duration** and **Status** are empty.





19.2 Duplicating campaigns

Duplicating a predefined and pre-configured campaign enables you to quickly reuse the code review context, without having to redefine the same fields multiple times. When a campaign is duplicated, the new campaign contains the same information as the original and is immediately ready to be launched. The selected code review and target cannot be edited.

 **Example**

When setting up the context of a code review process, you might need to launch multiple campaigns to make sure the code review process is carried out properly and checks the desired conditions.

You can create one campaign, using the desired code review and target. You can then duplicate the campaign as many times as needed, instead of manually creating all the campaigns.

To duplicate a campaign, either right-click on the item in the  **Campaigns** search view and select  **Duplicate**, or select the item and click the  Duplicate icon in the toolbar. The duplicated campaign is displayed in the list in the  **Campaigns** search view and is ready to be launched.

The duplicated campaign contains the same exact information as the original, except for the **Identifier** because it must always be unique. The code of the duplicated campaign is defined automatically depending on the code review selected to create the original campaign.

- If the **Campaign Identifier Template** defined in the code review contains the variable \$i, then the code of the duplicated campaign will reuse the same default code with the variable incremented by one.
- If the **Campaign Identifier Template** defined in the code review does not contain the variable \$i, then the code of the first duplicated campaign will be "*the original campaign's identifier (1)*". The code of the second duplicated campaign will be "*the original campaign's identifier (2)*", and so on.

19.3 Launching campaigns manually


After manually creating a campaign, follow the subsequent steps to manually launch the campaign and start the code review process.



 **Important!**

A campaign can only be manually launched *once*. To launch a second campaign created using the same code review, a new campaign must be manually created or duplicated, or a campaign schedule must be used to create new campaigns automatically.

 **Reference**

For more information about scheduling campaigns, refer to [Schedules on page 114](#).

Step 1 Locate the campaign in the  **Campaigns** search view.

Step 2 Either right-click on the item in the view and select  **Launch**, or select the item in the view and click the  Launch icon in the toolbar.


Result The campaign is launched and the code review process starts. The **Status** column is updated with the progress being made. The number of members checked is displayed to show how quickly the code review process is going.



When the code review process is finished and the campaign is completed, an automatic recap email is sent if the [Send Recap After Execution](#) box was checked in the code review's editor.

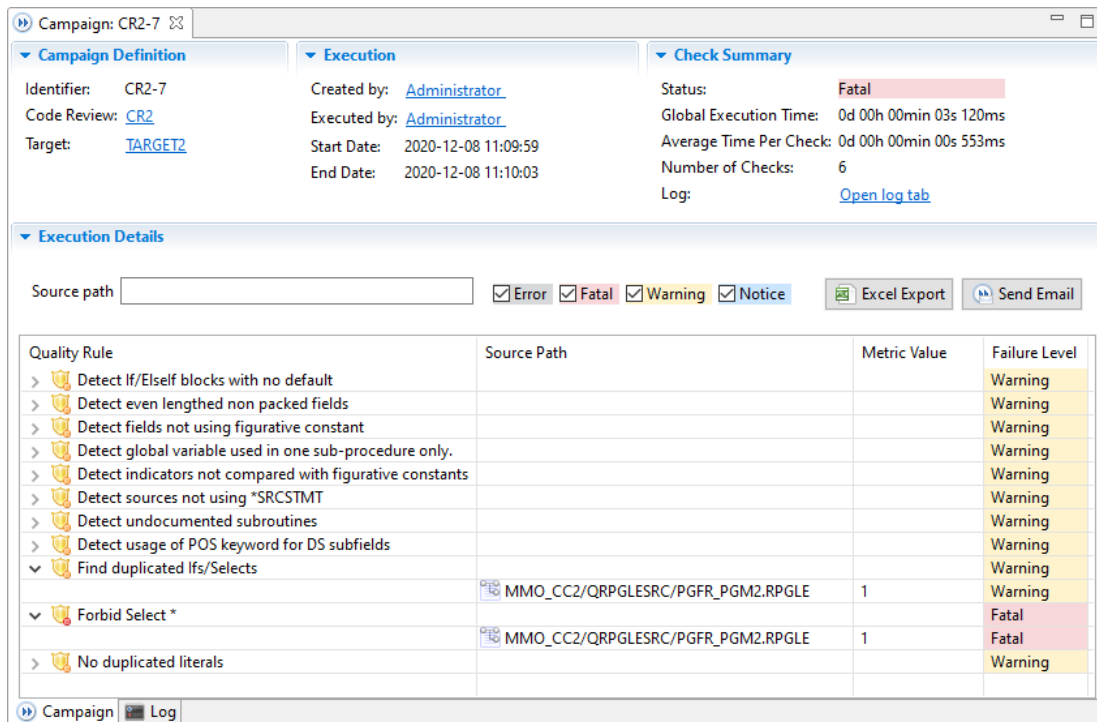
 **Note**

To see the results of any campaign, either manually launched or scheduled, open the campaign's editor.

19.4 Understanding campaign results

To open a campaign's result view and review the results of the finished code review process, locate the campaign in the  **Campaigns** search view, then either:

- double-click on the item in the view,
- right-click on the item in the view and select  **Open Campaign Results**, or
- select the item in the view and click the  Results icon in the toolbar.



The screenshot shows the 'Campaign Results' view for campaign CR2-7. It is divided into three main sections: Campaign Definition, Execution, and Check Summary.

- Campaign Definition:** Identifier: CR2-7, Code Review: CR2, Target: TARGET2.
- Execution:** Created by: Administrator, Executed by: Administrator, Start Date: 2020-12-08 11:09:59, End Date: 2020-12-08 11:10:03.
- Check Summary:** Status: Fatal, Global Execution Time: 0d 00h 00min 03s 120ms, Average Time Per Check: 0d 00h 00min 00s 553ms, Number of Checks: 6, Log: [Open log tab](#).

Below these sections is the 'Execution Details' section, which includes a 'Source path' input field and checkboxes for Error, Fatal, Warning, and Notice. There are also 'Excel Export' and 'Send Email' buttons.

Quality Rule	Source Path	Metric Value	Failure Level
> Detect If/Elseif blocks with no default			Warning
> Detect even lengthed non packed fields			Warning
> Detect fields not using figurative constant			Warning
> Detect global variable used in one sub-procedure only.			Warning
> Detect indicators not compared with figurative constants			Warning
> Detect sources not using *SRCSTMT			Warning
> Detect undocumented subroutines			Warning
> Detect usage of POS keyword for DS subfields			Warning
✓ Find duplicated lfs/Selects	MMO_CC2/QRPGLESRC/PGFR_PGM2.RPGLE	1	Warning
✓ Forbid Select *	MMO_CC2/QRPGLESRC/PGFR_PGM2.RPGLE	1	Fatal
> No duplicated literals			Warning

Figure 46: The Campaign Results view

Important!
Campaigns cannot be edited. The campaign's results view can only be used to view information about the code review process and the campaign results.

Note
The campaign results are ordered by their security level.

It is possible to open a quality rule's web documentation page from the **Campaign Results** view. To do so, you can either double click on the chosen quality rule to open its documentation in a browser, or right-click on the quality rule and select **Open Rule documentation**.

Campaign Definition

Identifier

This field displays the unique campaign's ID defined during the creation of the campaign.

Code Review

This field displays the code review selected during the creation of the campaign. Follow the link to access the code review's editor.

Target

This field displays the target selected during the creation of the target. Follow the link to access the target's editor.

Execution

Created by & Executed by: These fields display, respectively, the user who created and the user who executed the campaign.

Start Date & End Date: These fields display, respectively, the day and time when the campaign was launched and when it ended.

Check Summary

Status: This field displays the campaign status.

After the code review process is finished, the campaign status is either changed to **OK** or to one of the failure levels: **NOTICE**, **WARNING**, or **FATAL**.

- If the campaign status is OK, it means that none of the quality rules executed failed during the code review process.
- If the campaign status is one of the failure levels, it means that at least one of the quality rules executed failed during the code review process.

The failure level given to the quality rule determines the campaign status. If multiple quality rules with multiple failure levels failed during the code review process, the most critical failure level is used to determine the campaign status.

- If the quality rules failed to return any results, the status is **ERROR**. The cause of the error can be found in the campaign's log file.

Example

During a code review process, 5 NOTICE and 2 FATAL quality rules failed. The FATAL failure level is more critical than the NOTICE level, so the campaign overall status will be changed to FATAL.

The **Global Execution Time** displays the total amount of time it took the campaign to execute.

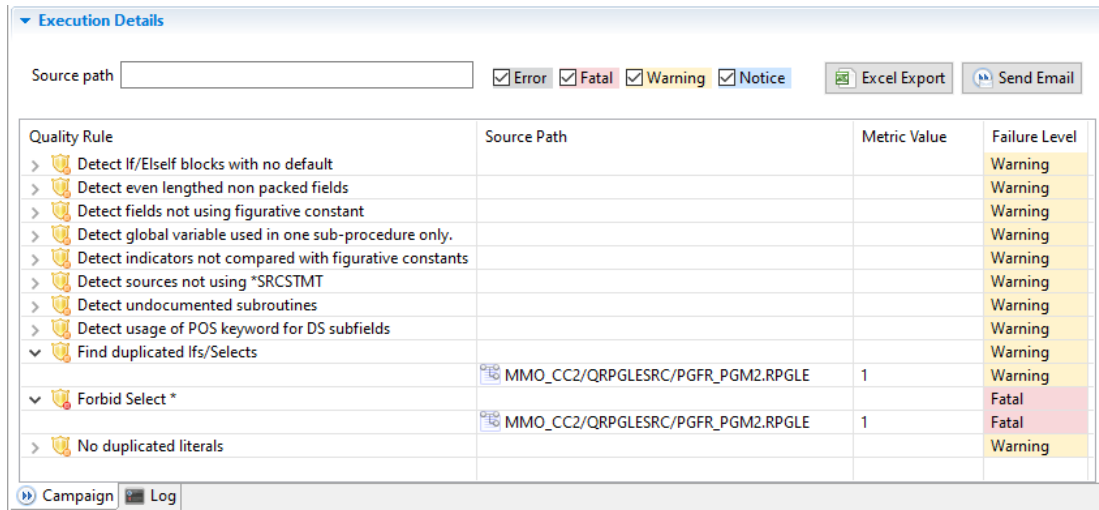
The **Average Time Per Check** displays the average time taken to check each source member during the campaign.

The **Number of Checked Source Members** displays the total number of source members checked during the campaign.

Log: Click the open log tab to view the campaign's logs. The log file opens in the **Log** tab.

Execution Details

This section displays detailed information about the code review process, and shows you precisely which quality rules fail and which source members are concerned.



Quality Rule	Source Path	Metric Value	Failure Level
> Detect If/Elseif blocks with no default			Warning
> Detect even lengthed non packed fields			Warning
> Detect fields not using figurative constant			Warning
> Detect global variable used in one sub-procedure only.			Warning
> Detect indicators not compared with figurative constants			Warning
> Detect sources not using *SRCSTMT			Warning
> Detect undocumented subroutines			Warning
> Detect usage of POS keyword for DS subfields			Warning
√ Find duplicated ifs/Selects	MMO_CC2/QRPGLESRC/PGFR_PGM2.RPGLE	1	Warning
√ Forbid Select *	MMO_CC2/QRPGLESRC/PGFR_PGM2.RPGLE	1	Fatal
> No duplicated literals			Warning

Figure 47: The Campaign Results execution details

To be able to navigate the results, a filter is available to display the results by **Target name** or **Status** (Error, Fatal, Warning, Notice, OK, Ignored).


Note

By default, only the Error, Fatal and Warning status are checked.

The **Quality Rule** column displays the complete list of rules included in the campaign. Each quality rule is associated with a **Status**:


- **OK** if the quality rule succeeded,
- the failure level given to the quality rule (**Notice**, **Warning**, or **Fatal**) if the quality rule failed at least once,
- **Error** if the quality rules failed to return any results. The cause of the error can be found in the campaign's log file.
- **Ignored** if the source was ignored by the code review process because the quality rules were not applicable (the source type is not supported by the quality rule).

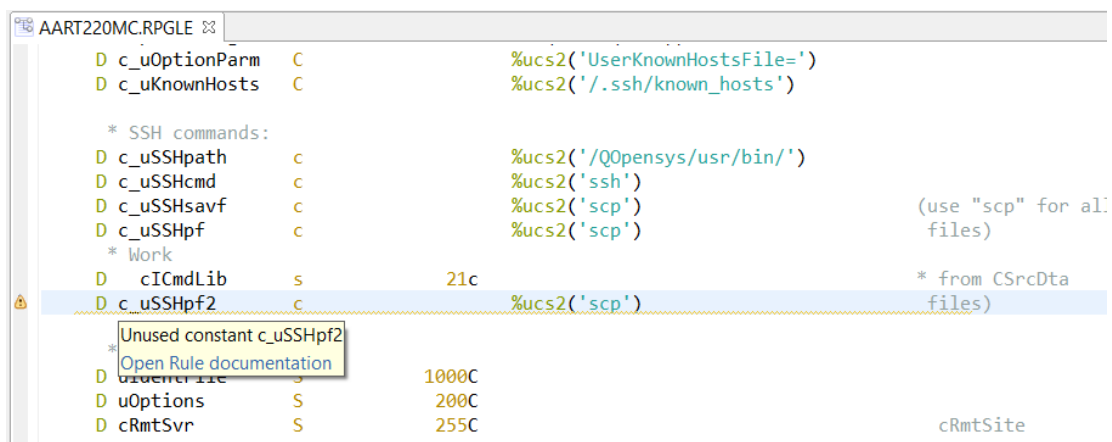
Expand a quality rule's node to display the list of all the source members that were checked during the code review process. For each source member:

- the **Source Path** column displays the source member's path,
- the **Metric Value** column displays the numeric value deduced from the source member's analysis. If an error occurs during the execution and no value could be deduced, a  warning icon is displayed. Information about the error is given in the tooltip and a full description is available in the logs.
- the **Status** column shows whether the quality rule succeeded or failed.

Source Code

To view a source member's source code in a dedicated view, locate the source member in the result view of the campaign, then either:

- double-click on the item in the view, or
- right-click on the item in the view and select  **Open Source**.



```

AART220MC.RPGLE
D c_uOptionParm C %ucs2('UserKnownHostsFile=')
D c_uKnownHosts C %ucs2('/.ssh/known_hosts')

* SSH commands:
D c_uSSHpath c %ucs2('/Q0pensys/usr/bin/')
D c_uSSHcmd c %ucs2('ssh')
D c_uSSHsavf c %ucs2('scp') (use "scp" for all files)
D c_uSSHpf c %ucs2('scp')
* Work
D cICmdLib s 21c * from CSrcDta
D c_uSSHpf2 c %ucs2('scp') files)
* Unused constant c_uSSHpf2
* Open Rule documentation
D uIdentFile S 1000C
D uOptions S 200C
D cRmtSvr S 255C cRmtSite
  
```

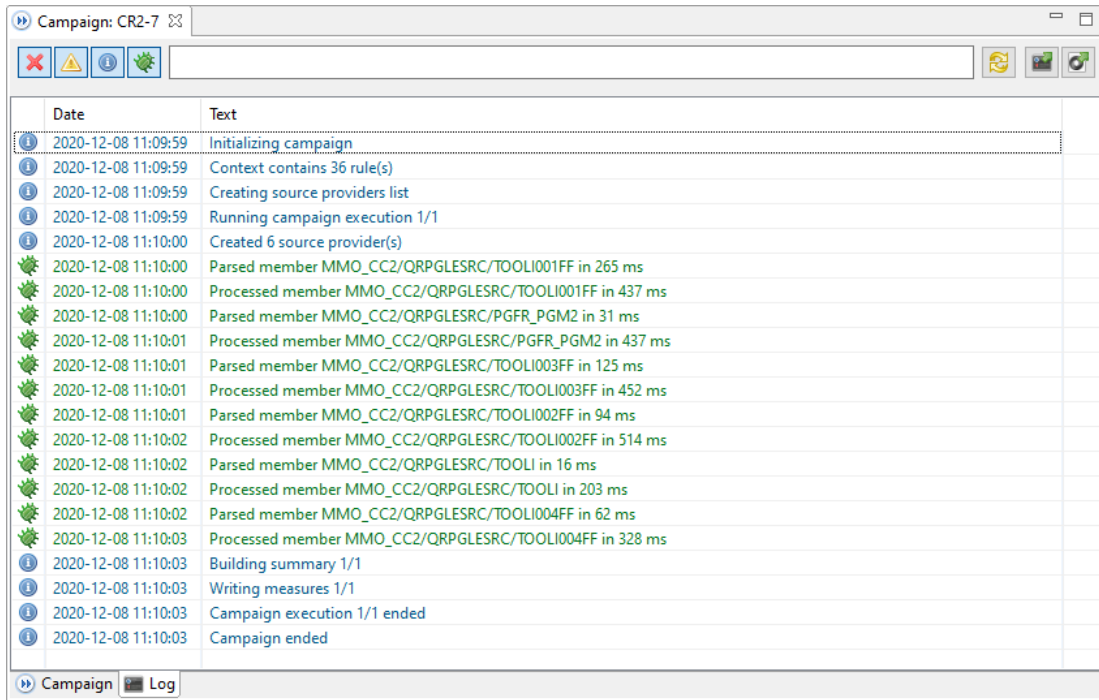
Figure 48: The source member's code view

The lines of code that do not comply with any of the rules in the rule set are underlined. On the left, a status icon gives the failure level of the source code.

To open a rule's documentation associated with non-compliant code, hover the mouse over it and click the link in the hint box to the documentation of the rule.

Log

This section displays detailed information about the campaign logs, and allows you to export the logs in text or JSON format.

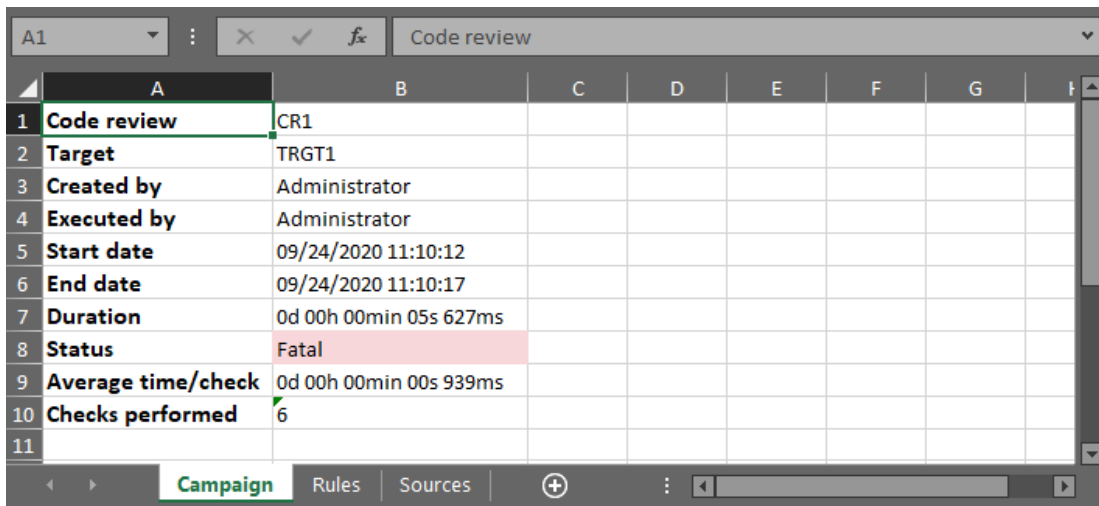


Date	Text
2020-12-08 11:09:59	Initializing campaign
2020-12-08 11:09:59	Context contains 36 rule(s)
2020-12-08 11:09:59	Creating source providers list
2020-12-08 11:09:59	Running campaign execution 1/1
2020-12-08 11:10:00	Created 6 source provider(s)
2020-12-08 11:10:00	Parsed member MMO_CC2/QRPGLESRC/TOOLI001FF in 265 ms
2020-12-08 11:10:00	Processed member MMO_CC2/QRPGLESRC/TOOLI001FF in 437 ms
2020-12-08 11:10:00	Parsed member MMO_CC2/QRPGLESRC/PGFR_PGM2 in 31 ms
2020-12-08 11:10:01	Processed member MMO_CC2/QRPGLESRC/PGFR_PGM2 in 437 ms
2020-12-08 11:10:01	Parsed member MMO_CC2/QRPGLESRC/TOOLI003FF in 125 ms
2020-12-08 11:10:01	Processed member MMO_CC2/QRPGLESRC/TOOLI003FF in 452 ms
2020-12-08 11:10:01	Parsed member MMO_CC2/QRPGLESRC/TOOLI002FF in 94 ms
2020-12-08 11:10:02	Processed member MMO_CC2/QRPGLESRC/TOOLI002FF in 514 ms
2020-12-08 11:10:02	Parsed member MMO_CC2/QRPGLESRC/TOOLI in 16 ms
2020-12-08 11:10:02	Processed member MMO_CC2/QRPGLESRC/TOOLI in 203 ms
2020-12-08 11:10:02	Parsed member MMO_CC2/QRPGLESRC/TOOLI004FF in 62 ms
2020-12-08 11:10:03	Processed member MMO_CC2/QRPGLESRC/TOOLI004FF in 328 ms
2020-12-08 11:10:03	Building summary 1/1
2020-12-08 11:10:03	Writing measures 1/1
2020-12-08 11:10:03	Campaign execution 1/1 ended
2020-12-08 11:10:03	Campaign ended

Figure 49: The Campaign Results log tab

19.5 Exporting campaign results

When a campaign is completed, you can export the campaign to *Excel* format. In the exported file, the **Campaign** tab gives the summary of the campaign and its execution details. The **Rules** tab lists all the rules that were applied during the campaign and the number of cases found. The **Sources** tab shows all the code review results on the sources.



	A	B	C	D	E	F	G	H
1	Code review	CR1						
2	Target	TRGT1						
3	Created by	Administrator						
4	Executed by	Administrator						
5	Start date	09/24/2020 11:10:12						
6	End date	09/24/2020 11:10:17						
7	Duration	0d 00h 00min 05s 627ms						
8	Status	Fatal						
9	Average time/check	0d 00h 00min 00s 939ms						
10	Checks performed	6						
11								

Figure 50: The campaign Excel export file

To export a campaign, either right-click on the item in the **Campaigns** search view and select **Export to Excel**, or select the item and click the **Excel Export** icon in the toolbar.




Note

To export campaigns to Excel on Linux-based systems, the *fontconfig* package must be installed on the machine.

19.6 Emailing campaign results

When a campaign is completed, an automatic recap email is sent if the [Send Recap After Execution](#) box was checked in the code review's editor. In addition, it is possible to manually send that recap email.


The recap email contains the same information as the results displayed in the campaign's editor. The email addresses to which the email is sent are defined in the **Emails** field in the code review's editor.

To manually send a recap email for a campaign, either right-click on the item in the  **Campaigns** search view and select  **Send Email**, or select the item and click the  Send icon in the toolbar.




19.7 Deleting campaigns

 **Warning!**

Deleted campaigns cannot be recovered.

The campaign execution results and the campaign logs are discarded.
The issues generated at the end of the execution are kept in the  **Issues** view.

Deleting a campaign that is still in execution will abort the execution. In that case, the issues found so far are not generated.

To delete a campaign, either right-click on it in the  **Campaigns** search view and select  **Delete**, or select it and click the  Delete icon in the toolbar. Click **OK** to confirm or click **Cancel** to keep the campaign.

20 Schedules

Required role	Campaign Execution
Access	CodeChecker Server → Campaign Execution → Schedules

Chapter Summary

- 20.1 Creating schedules 114
- 20.2 Editing schedules 115
- 20.3 Starting and stopping schedules 119
- 20.4 Understanding scheduled campaign results 119
- 20.5 Deleting schedules 119

Schedules are used to automatically create and launch campaigns, at a specified date and time or at a specified interval of time. Schedules can be defined to create one campaign at a later date, or to create multiple campaigns over time, following a recurring schedule.

Reference

This chapter describes how to create and manage campaign schedules. For more information about manually creating and launching campaigns, refer to [Campaigns on page 105](#).

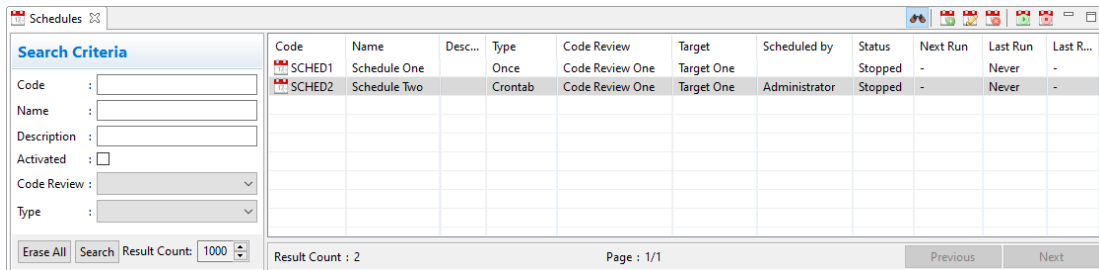


Figure 51: Schedules

The **Schedules** view is accessed from the **Campaign Execution** node in the **Navigator**.

Enter any combination of the above search criteria, then click the **Search** button to display the results. To display the complete list, click the **Search** button without entering any search criteria.

If a schedule has been started, its **Status** is **Started**. If a schedule has not been started yet, has been stopped or is finished, its **Status** is **Stopped**.

The **Next Run** column displays the next date and time the schedule will be executed. The **Last Run** column displays the most recent date and time the schedule was executed.

The **Last Result** column displays the most recent campaign status. The campaign status can either be OK or one of the failure levels: NOTICE, WARNING, or FATAL.

20.1 Creating schedules

Follow the subsequent steps to create a new schedule.

Step 1 To access the **Create Schedule** wizard, either click the **Create** icon in the toolbar of the **Schedules** search view, or right-click anywhere in the list and select **Create Schedule**.

Step 2 Define the schedule's **Code** and **Name**. These values are required to create a new schedule. The **Code** cannot be changed once the schedule is created, but the **Name** can be edited later.


Step 3 Select a **Code Review** from the drop-down list. This value is required and cannot be edited once the schedule is created.

When the scheduled campaign is launched, the rule sets defined in the selected code review will be executed on the associated target.

Step 4 Select a **Target** from the drop-down list. This value is required and cannot be edited once the schedule is created.

When the scheduled campaign is launched, the code review will be carried out on the selected target application.


Click **Finish**.



Result The new schedule is created and its editor opens automatically. It is displayed in the search list in the  **Schedules** search view.

 **Important!**

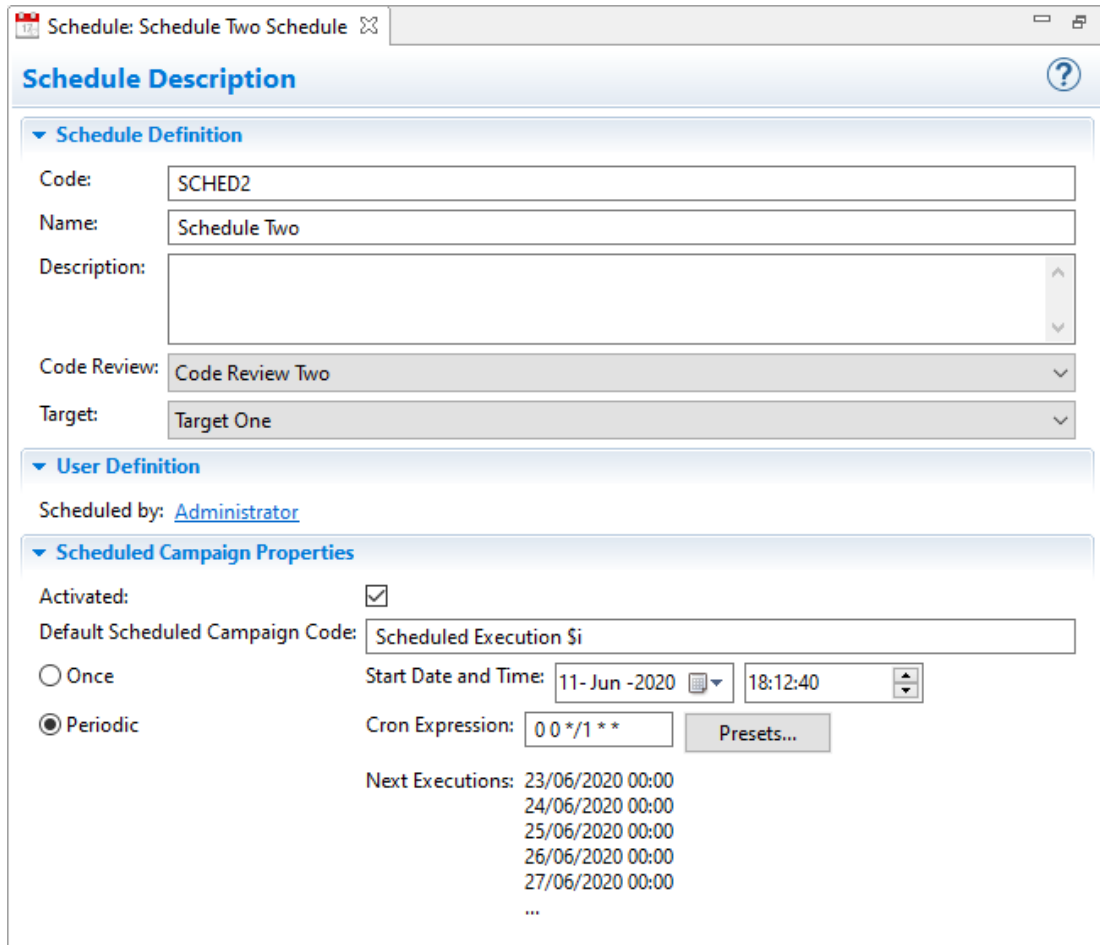
It is required to edit new schedules before using them.

20.2 Editing schedules

To open a schedule's editor, locate the schedule in the  **Schedules** search view, then either:

- double-click on the item in the view,
- right-click on the item in the view and select  **Edit**, or
- select the item in the view and click the  Edit icon in the toolbar.

Save the changes (, Ctrl+S or **File > Save**).



The screenshot shows a window titled "Schedule: Schedule Two Schedule". The main content area is titled "Schedule Description" and contains three expandable sections:

- Schedule Definition:** Includes fields for Code (SCHED2), Name (Schedule Two), Description (empty), Code Review (Code Review Two), and Target (Target One).
- User Definition:** Shows "Scheduled by: Administrator".
- Scheduled Campaign Properties:** Includes an "Activated" checkbox (checked), a "Default Scheduled Campaign Code" field (Scheduled Execution \$i), radio buttons for "Once" and "Periodic" (selected), a "Start Date and Time" field (11-Jun -2020 18:12:40), a "Cron Expression" field (0 0 */1 **), a "Presets..." button, and a list of "Next Executions" (23/06/2020 00:00, 24/06/2020 00:00, 25/06/2020 00:00, 26/06/2020 00:00, 27/06/2020 00:00, ...).

Figure 52: The Schedule editor

Schedule ID

Code

This field displays the schedule's unique ID defined when the schedule was created. Once a schedule is created, it is not possible to edit this code.

Name

The schedule's name should reflect the context of the campaigns that will automatically be created and launched to be easily identified.

Description

Use this description to give as many details as possible about the type of schedule used to create campaigns.

Code Review

This field displays the code review selected when the schedule was created.

Target

This field displays the target selected when the schedule was created.

Scheduled by

This link displays the user that created the scheduled campaign. Click the link to see the user profile.

Scheduled Campaign Properties

Activated

When this box is checked, the schedule will restart automatically every time the CodeChecker Server starts.

Schedules must be started once to be considered active and to start running. If the CodeChecker Server is stopped, all the schedules will be stopped as well. When the server is started again, schedules with the **Activated** box checked will automatically start and run accordingly, without having to open the CodeChecker Studio and manually start them.



Reference

For more information about starting and stopping schedules, refer to [Starting and stopping schedules on page 119](#).

Default Scheduled Campaign Code

Define the default code to assign to every campaign created by the schedule, if it is set to repeat. This code will be the scheduled campaign's ID.

This default code should clearly reflect the schedule to make tracking each of the campaigns it creates easier, and to easily identify which campaigns were created by a schedule and which were created manually.

Use the variable \$i in this field to track the number of scheduled campaigns created. The variable is incremented by one every time a new scheduled campaign is created by the schedule, if it is set to repeat.

Type

Select the type of schedule to run from the drop-down list.

Each type has a different set of parameters to define. Use the [Scheduled Campaign Execution Information](#) field on the right to ensure the defined parameters are correct.

Once

Launch the scheduled campaign one time. Select a date and time in the future to automatically launch the scheduled campaign.

Periodic

Launch scheduled campaigns following an interval of time defined by a **Cron Expression**. Enter a cron expression in the field to create a scheduling pattern or select one of the presets cron expressions. Once the schedule is started, the campaign will be launched when the parts in its scheduling pattern will be true at the same time.

The frequency is defined by crontab-like rules.

- **Minutes sub-pattern:** on which minute of the hour should the task be launched? The values range from 0 to 59.
- **Hours sub-pattern:** on which hour of the day should the task be launched? The values range from 0 to 23.
- **Days of month sub-pattern:** on which days of the month should the task be launched? The values range from 1 to 31. The special value "L" can be used to recognize the last day of the month.
- **Months sub-pattern:** during which months of the year should the task be launched? The values range from 1 (January) to 12 (December). The aliases jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov and dec can also be used.
- **Days of week sub-pattern:** on which days of the week should the task be launched? The values range from 0 (Sunday) to 6 (Saturday). The aliases sun, mon, tue, wed, thu, fri and sat can also be used.

Note

The asterisk wild-card character * is also recognized, indicating "every minute of the hour", "every hour of the day", "every day of the month", "every month of the year" and "every day of the week", according to the sub-pattern in which it is used.

Reference

For more information about this scheduling pattern, refer to this website

<http://www.sauronsoftware.it/projects/cron4j/manual.php>.

Scheduled Campaign Execution Information


This box displays a translation in real text of the parameters defined for the [Type](#) of schedule. If there is an error, the box turns red to indicate that the schedule is invalid.



Next Scheduled Campaigns (max. 5)

This field displays the date and time of the next five scheduled campaigns, if it is set to repeat and depending on the defined schedule.

20.3 Starting and stopping schedules

After creating a schedule, it must be started before the first scheduled campaign can be created and launched. Follow the subsequent steps to start a schedule.




Step 1 Locate the schedule in the  **Schedules** search view.

Step 2 Either right-click on the item in the view and select  **Start**, or select the item in the view and click the  Start icon in the toolbar.

Click **OK**.

Result The schedule is started. When the date and time defined in the schedule arrive, a new campaign will be automatically created and launched.

If they are set to repeat, started schedules will continue to create new campaigns and launch them until they are stopped.

To stop a schedule, either right-click on the item in the  **Schedules** search view and select  **Stop**, or select the item and click the  Stop icon in the toolbar. Click **OK** to confirm.

 **Note**

Schedules can be started and stopped as often as required.

20.4 Understanding scheduled campaign results

After starting a schedule, new campaigns are automatically created and launched when the date and time defined in the schedule arrive. To see the results of any campaign, either scheduled or manually launched, open the campaign's editor.

 **Reference**




For more information about accessing and understanding campaign results, refer to [Understanding campaign results on page 108](#).

20.5 Deleting schedules

 **Warning!**

Deleted schedules cannot be recovered.

The campaigns that were created by a parent schedule are not deleted.

To delete a schedule, either right-click on it in the  **Schedules** search view and select  **Delete**, or select it and click the  Delete icon in the toolbar. Click **OK** to confirm or click **Cancel** to keep the schedule.

21 Issues

Required role	Issues Management
Access	CodeChecker Server → Issues

Issues are the individual results of the quality evaluation of the source code. They are created when a campaign is launched and are then stored on the server independently from the campaign that uncovered them. When the campaign is re-executed, the issues' delta is managed as follows:

- new detected issues are created on the server,
- existing detected issues are updated on the server,
- fixed issues (no longer detected) are removed from the server.

Important!
 For ARCAD targets, the first execution of the campaign checks all the repository sources (closed versions only), and the next executions of the campaign check version sources only.

Reference
 For more information about campaigns, refer to [Campaigns on page 105](#).

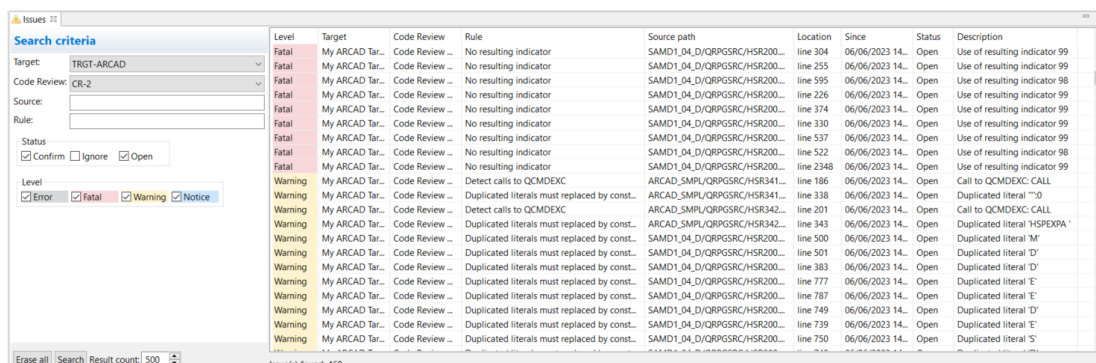


Figure 53: Issues

The **Issues** view is accessed directly from the **Navigator**.

It shows all the issues found in the campaigns. You can filter the issues by their level of severity: **FATAL**, **ERROR**, **WARNING**, or **NOTICE**, their status: **Open**, **Ignore**, or **Confirm**; or use the dynamic filter to display specific issues according to their Rule, Code Review, etc.

Note
 Results with **OK** or **Ignored** security levels are not listed in the **Issues** view.

Click the refresh icon to reload all issues from the CodeChecker Server.

Reviewing issues

Issues are created when a campaign is launched and are then stored on the server independently from the campaign. The **Since** column gives the date when the issue was first discovered. This information is kept even if a new campaign execution detects the same issue again.

The **Location** column gives the line number where the issue is situated in the source file. When the location is **Undefined**, it means that the issue applies to the source file as a whole.

Right-click on an issue and select an option in the contextual menu to qualify the status of the issue:

- **Open**: for issues that have not been reviewed. By default, all issues are **Open**.
- **Ignore**: for issues that require no actions.
- **Confirm**: for issues that require action.

Level	Target	Code Review	Rule	Source pat
Fatal	My ARCAD Tar...	Code Review ...	No resulting indicator	ARCAD_SM
Fatal	My ARCAD Tar...	Code Review ...	No resulting indicator	ARCAD_SM
Fatal	My ARCAD Tar...	Code Review ...	Forbid CALLS without PLIST	ARCAD_SM
Fatal	My ARCAD Tar...	Code Review ...	Forbid CALLS without PLIST	ARCAD_SM
Fatal	My ARCAD Tar...	Code Review ...	No resulting indicator	ARCAD_SM
Fatal	My ARCAD Tar...	Code Review ...	No resulting indicator	ARCAD_SM
Fatal	My ARCAD Tar...	Code Review ...	Forbid GOTO statements	SAMD1_04
Fatal	My ARCAD Tar...	Code Review ...	Forbid GOTO st...	SAMD1_04
Fatal	My ARCAD Tar...	Code Review ...	Forbid not keye...	SAMD1_04
Fatal	My ARCAD Tar...	Code Review ...	Forbid not keye...	SAMD1_04
Fatal	My ARCAD Tar...	Code Review ...	Forbid not keyed file access	SAMD1_04
Fatal	My ARCAD Tar...	Code Review ...	Forbid not keyed file access	SAMD1_04
Fatal	My ARCAD Tar...	Code Review ...	Forbid not keyed file access	SAMD1_04

Issue(s) found: 460

Figure 54: Review issues

The status of the issue is changed in the view.

Double-click on an issue to open the source file in the CodeChecker Studio. The issues are highlighted in the source code view.

Note

It is also possible to open a rule's documentation from the source code view. To do so, hover the mouse over an issue to get the hyperlink to the documentation of the issue's rule.

Important!

Issues with status **Ignore** are not highlighted in source code view.

22 The Command Line Interface

The ARCAD CodeChecker Command Line Interface executes campaigns of code quality tests from the command line. Use the CLI to execute predefined campaigns from a command line.

The ARCAD CodeChecker CLI is distributed as a zip file. Extract the content into the folder of your choice. Go to the `/bin` folder to launch the CLI.

Note

The CLI needs to have access to a Java 8 Runtime Environment that needs to be installed and accessible from anywhere in a command line terminal (run `<code>java -version </code>` to find the current accessible Java version).

CodeChecker CLI for IBM i

The CodeChecker CLI is also available as an RPM package for IBM i platforms, that can be installed using **rpm** or **yum**.

Before you start, open a terminal session and run the **yum version** command to make sure everything is correctly installed in your environment.

It is recommended to adjust the path to use the absolute path of programs, so you do not need to specify it manually. In this particular case, it allows you to use `codechecker` instead of `/QOpenSys/pkg/bin/codechecker` to run a CodeChecker CLI command.

Reference

For more information about RPM packages, refer to the [installation](#) and [path management](#) documentations.

Follow the subsequent steps to install the CodeChecker CLI on IBM i:

Step 1 Copy the **codecheckerCLI-X.Y.Z.ibm7.2.ppc64.rpm** file and paste it in any folder on the IFS of the IBM i target.

Step 2 Open a terminal session and locate yourself in the folder where the rpm file is stored.

Step 3 Run the **yum install codecheckerCLI-X.Y.Z.ibm7.2.ppc64.rpm** command to install the package.

Step 4 Enter **y** and press enter to confirm.

Result The new CodeChecker CLI is successfully installed.

To remove the CodeChecker CLI, open a terminal session on the target IBM i and run the **yum remove codecheckerCLI** command. Enter **y** and press enter to confirm.

To call any action using the CLI, the following syntax is used: `codechecker <subcommand> <parameters>`.

If no subcommand is entered, the CLI then displays a list of the subcommands available.

Each subcommand requires one or more parameters. To display the list of supported parameters for any subcommand, run the subcommand without any parameters. If nothing is provided, the system prompts you to fill in the blanks required to accomplish the subcommands tasks.

Parameters between *[brackets]* are optional. Each parameter has a long and a short form (example: `--server`, `-S`) and are separated from their value by the equal sign `=` or a space.

There are two subcommands available in the CodeChecker CLI:

- [Execute Campaign \(executeCampaign\)](#)
- [Execute ARCAD Campaign \(executeArcadCampaign\)](#)
- [Execute external campaign \(executeLocalCampaign\)](#)

Common parameters

The following parameters are common to all CLI commands.

-C, --crypted

Is used if the password is Base 64 encrypted.

-H, --help

Displays the help message for the command.

-L, --log-file

Indicates the path to the file where the logs are written.

-P, --password

Indicates the password used to connect to the server.

-S, --server

Indicates the URL to the server to which to connect.

-U, --user

Indicates the user used to connect to the server.

-V, --verbose

Enables the **DEBUG** and **TRACE** log levels.

Execute Campaign (executeCampaign)

This subcommand executes a campaign.

Usage: codechecker executeCampaign [-CHrV] -P[=<password>] [-P[=<password>]]...-

c=<codeReviewCode> [-f=<failOnStatus>][-L=<logFile>] [-m=<maxLogLevel>][-

o=<campaignSummaryOutput> -S=<serverURL>-t=<targetCode> [-u=<issuesListOutput>]-U=<user>

The available parameters for the **executeCampaign** subcommand are the following:

-c, --codereview

Indicates a Code Review code.

-f, --fail-status

Indicates the Campaign status for which the execution fails. Possible values are **NOTICE**, **WARNING**, **FATAL** or **ERROR**.

-m, --max-level

Indicates the maximum level of log to display. Possible values are **DEBUG**, **INFO**, **WARNING** or **ERROR**.

-n, --fail-on-error

The execution of a campaign via the CLI is set to fail if the campaign encounters error(s) during the process.

-o, --summary-output

Indicates the file where the JSON campaign summary must be written.

-p, --sonar-project

Indicates the folder to create a SonarQube project (issues report + source). The content of that directory can then be used to invoke the Sonar Scanner CLI to upload the campaign results in SonarQube.

-r, --reset-target

Resets a specified target and clears existing issues before executing the campaign.

-s, --ibmi-server

Indicates an IBM i server code, as defined in ARCAD CodeChecker.

-t, --target

Indicates the target code.

-u, --issues-output

Indicates where the JSON issues list are written. The value expected is a file path.

Execute ARCAD Campaign (executeArcadCampaign)

This subcommand executes an ARCAD campaign.

Usage: codechecker executeArcadCampaign [-CHqV] -P[=<password>] [-P [=<password>]]... -a=<applicationCode> -c=<codeReviewCode> -e=<environmentId> [-f=<failOnStatus>] -i=<arcadInstance> -l=<arcadLanguage> [-L=<logFile>] [-m=<maxLogLevel>] [-o=<campaignSummaryOutput>] -s=<serverCode> -S=<serverURL> -U=<user> -v=<versionNumber>

The available parameters for the executeArcadCampaign subcommand are the following:

-a, --application

Indicates an ARCAD application code.

-c, --codereview

Indicates a Code Review code.

-e, --environment

Indicates an ARCAD application environment code.

-d, --arcad-list

Indicates the path of an ARCAD list that gathers the campaign outputs. You can either set the path by its name or by using a LIBRARY/NAME format.

Once launched, the ARCAD list emptied if it exists and then filled with one entry per source member retrieved during the campaign execution.

**Important!**

This parameter is supported only when you run the CLI on IBM i.

-f, --fail-status

Indicates the Campaign status for which the execution fails. Possible values are **NOTICE**, **WARNING**, **FATAL** or **ERROR**.

-i, --instance

Indicates an ARCAD instance code.

-l, --language

Indicates an ARCAD language code (it can be ENG or FRA).

-m, --max-level

Indicates the maximum level of log to display. Possible values are **DEBUG**, **INFO**, **WARNING** or **ERROR**.

-n, --fail-on-error

The execution of a campaign via the CLI is set to fail if the campaign encounters error(s) during the process.

-o, --summary-output

Indicates the file where the JSON campaign summary must be written.

-p, --sonar-project

Indicates the folder to create a SonarQube project (issues report + source). The content of that directory can then be used to invoke the Sonar Scanner CLI to upload the campaign results in SonarQube.

-q, --sonar

Is used to upload the campaign result to SonarQube (if the SonarQube connection is configured).

-r, --reset-target

Resets a specified target and clears existing issues before executing the campaign.

-s, --ibmi-server

Indicates an IBM i server code, as defined in ARCAD CodeChecker.

-u, --issues-output

Indicates where the JSON issues list are written. The value expected is a file path.

-v, --version

Indicates an ARCAD application version number.

Execute external campaign (executeLocalCampaign)

This subcommand allows sources to be uploaded on the CodeChecker Server from an external location and the creation of an External Target type.

Usage: codechecker executeLocalCampaign [-CHnqrV] -P[=<password>]-c=<codeReviewCode>[-d=<targetDescription>][-f=<failOnStatus>] [-L=<logFile>][-m=<maxLogLevel>][-o=<campaignSummaryOutput>][-p=<sonarProjectOutput>][-s=<ibmiServer>] -S=<serverURL>-t=<targetCode> [-u=<issuesListOutput>]-U=<user> [-x=<deltaPath>][-z=<splitSize>] <mainPath><mainPath> Root source folder

The available parameters for the **executeLocalCampaign** subcommand are the following:

-c, --codereview

Indicates a Code Review code.

-d, --description

Indicates the target's description.

-f, --fail-status

Indicates the Campaign status for which the execution fails. Possible values are **NOTICE**, **WARNING**, **FATAL** or **ERROR**.

-m, --max-level

Indicates the maximum level of log to display. Possible values are **DEBUG**, **INFO**, **WARNING** or **ERROR**.

-n, --fail-on-error

Indicates whether the CLI execution fails if the campaign contains errors.

-o, --summary-output

Indicates the file where the JSON campaign summary must be written.

-p, --sonar-project

Indicates the folder to create a SonarQube project (issues report + source). The content of that directory can then be used to invoke the Sonar Scanner CLI to upload the campaign results in SonarQube.

-q, --sonar

Is used to upload the campaign result to SonarQube (if the SonarQube connection is configured).

-r, --reset-target

Resets a specified target and clears existing issues before executing the campaign.

-s, --ibmi-server

Indicates an IBM i server code, as defined in ARCAD CodeChecker.

-t, --target

Indicates a target code. If the target does not exist, it is created with this value set in this code parameter.

-u, --issues-output

Indicates where the JSON issues list are written. The value expected is a file path.

-x, --delta-folder

Indicates the root source delta folder.

-z, --split-size

Indicates, in kilobytes, the size used to split the source package sent to the server. By default, the split size is set to 5120 Kb.



ARCAD

CODECHECKER PLUG- INS

Introduction to the ARCAD CodeChecker plug-ins

The entire code review process can be carried out from the CodeChecker Studio. However, ARCAD CodeChecker can also be used by different platforms, via plug-ins.

These plug-ins are easy and quick to install, and offer additional features for the code review process.

All of the following plug-ins are accessed and managed inside their respective platforms.

- [ARCAD CodeChecker for RDi on page 129](#)
- [ARCAD CodeChecker for Jenkins on page 139](#)
- [ARCAD CodeChecker for SonarQube on page 146](#)

23 ARCAD CodeChecker for RDi

Chapter Summary

23.1 Connecting to the CodeChecker Server.....	129
23.2 Accessing the RDi plug-in.....	131
23.3 Setting rules execution preferences.....	132
23.4 Displaying rule sets.....	133
23.5 Executing quality rules.....	134
23.6 Managing executions.....	136
23.7 Managing error lists.....	137

The ARCAD CodeChecker plug-in for RDi is designed for the RDi integrated development environment (IDE).

The plug-in is designed for developers working in the RDi IDE. It enables developers to test the quality of the source code they are working on by executing active rule sets or specific quality rules, and know instantly if the source code does not comply with the standards set for the development team. With the plug-in, developers can easily know which quality rules are failing and identify which part of their source code is causing the failure. The portions of code causing the failures are clearly identified, making it easy for developers to modify their source code to comply with the quality standards.

23.1 Connecting to the CodeChecker Server

The ARCAD CodeChecker plug-in for RDi must be connected to the CodeChecker Server to access and use all the elements required for carrying out a code review process.

 **Warning!**

The plug-in cannot be used if it is not connected to the CodeChecker Server.

Follow the subsequent steps to connect to the CodeChecker Server.

Step 1 To access the **Preferences** window, open the **Window** menu then select **Preferences**.

Step 2 Expand the **ARCAD CodeChecker** node on the left-hand side of the window and select **Connection**.

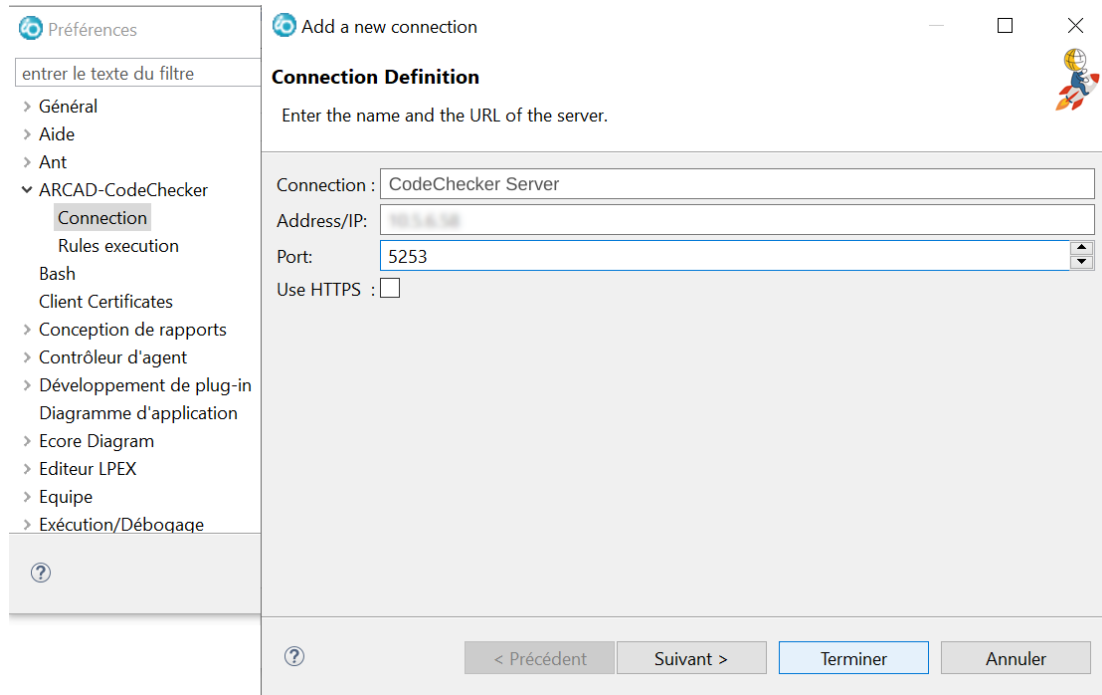


Figure 55: The CodeChecker Server connection preferences in RDi

Step 3 Define the following fields:

Connection

Sets a unique name for the connection to the CodeChecker Server.

Address/IP

Sets the *http*, *https* or *IP* address used to connect to the CodeChecker Server

Port

Sets the port number used to connect to the CodeChecker Server. By default, the CodeChecker Server uses the 5253 port number or the 52530 port number for the secured connection (HTTPS), but these numbers may have been changed during configuration.

Use HTTPS

Enable this option to use a secure connection.

Step 4 Click **Test connection** to make sure the connection is successful after entering your ARCAD CodeChecker credentials to access the server.

Step 5 To save any changes made, click **Apply**.

Result The plug-in is connected to the CodeChecker Server and is ready to be used.

Note

If the plug-in is not connected to the CodeChecker Server and you try to use one of its features, a message appears to inform you that a connection to the server is required.

Click **Yes** to access the **Server Configuration** menu and connect to the server, or **No** to cancel the operation.

23.1.1 Editing TLS Settings

Editing the TLS settings makes it possible to specify the Trust Store to use when connecting to CodeChecker Server through secure URL (*https*).

Follow the subsequent steps to redefine the security store parameters.

Step 1 Click on the **Edit TLS Settings** button.

Step 2 Edit the absolute paths to the **Trust-** and **KeyStore** as well as the passwords required to access and modify the files to use your own keys.

	Trust Store Properties	Key Store Properties
File Name	The absolute path to the TrustStore file.	The absolute path to the KeyStore file.
Password	The password required to access to this file. The default password for the ARCAD key file is quadra .	The password required to access to this file. The default password for the ARCAD key file is quadra .

Table 5: The TLS Settings

Step 3 Click the **Import Certificate** button to import your own certificates into the file indicated in the **TrustStore** path.

Step 4 Click **OK** to save your changes or **Cancel** to keep the default settings.

Result The plug-in is connected to the CodeChecker Server and is ready to be used.

Note


The **Reset** button resets the security store file information to the default values provided by ARCAD.

23.2 Accessing the RDi plug-in

The ARCAD CodeChecker plug-in for RDi can be accessed two ways in RDi:

1. from the **Remote Systems** view of the **Remote System Explorer** perspective,
2. within an i Project in the **i Projects** perspective.

The plug-in does not have its own perspective in RDi: the features are accessed from the contextual menu.

To access the plug-in, right-click on a program file to open the contextual menu. The features are available in the  **ARCAD CodeChecker** menu.

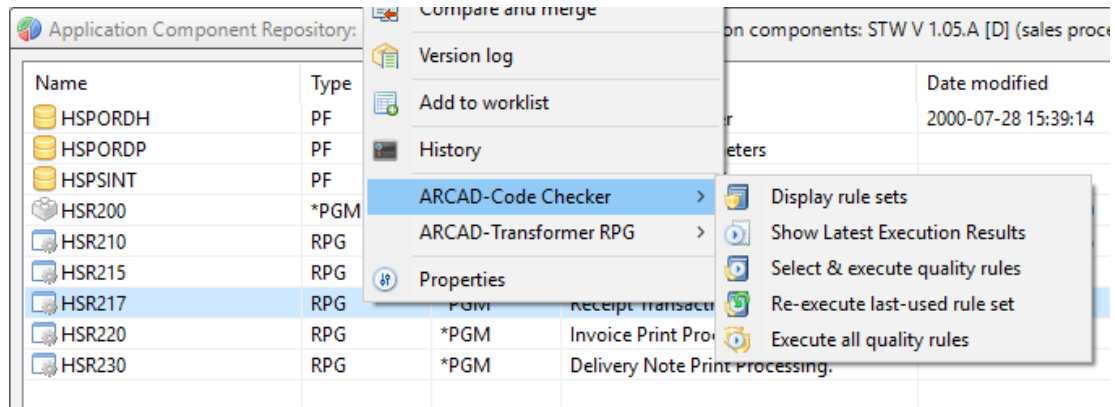



Figure 56: The ARCAD CodeChecker plug-in for RDi

Important!
The plug-in is only available for source members. It is not available if you open the contextual menu for another type of file.

23.3 Setting rules execution preferences

Important!
These preferences options are specific to the RDi studio. The  Configuration Management role is required for this task.

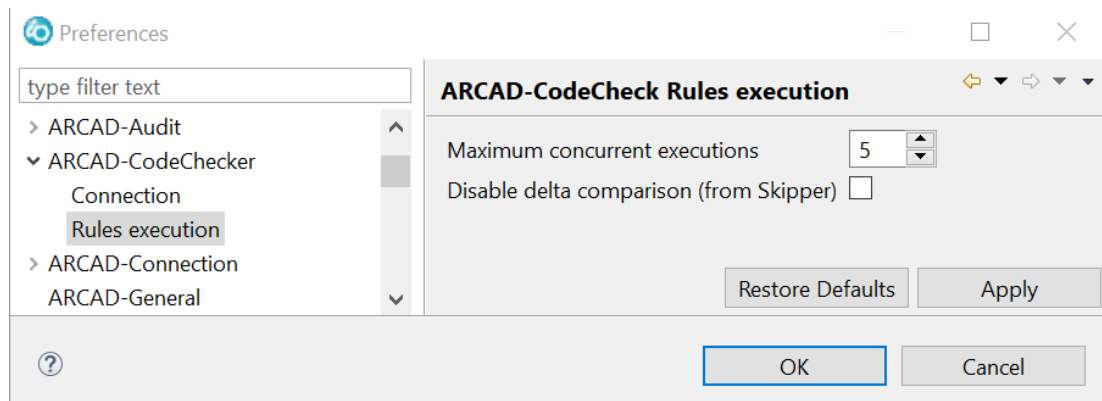


Figure 57: The Preference dialog in RDi - Rules Execution

To access the **Preferences** window, open the **Window** menu then select **Preferences**. Open the **Rules execution** settings.

Maximum concurrent executions

Set the maximum number of concurrent executions of rules in ARCAD CodeChecker. When executing with the ARCAD CodeChecker plug-in for RDi, executions of the rules will be queued.

Disable delta comparison (from ARCAD Skipper)

By default, when running an execution on ARCAD Skipper components, a delta comparison process is applied on issues found in the component in version and issues found in the component in the application repository. If the issues are found in both versions, they are ignored.

Tick this checkbox to disable this comparison process when running executions on ARCAD Skipper.

Important!

This parameter does not apply to executions performed from the **Remote System Explorer**.

23.4 Displaying rule sets

The **Quality rules** used to analyze the source code of an application are grouped into rule sets in the CodeChecker Studio.




Use rule sets to organize quality rules in a logical way: by language, by theme, and so on. The standard quality rules are already organized into several generic rule sets. You can create new rule sets to manage and categorize your custom quality rules, or re-organize the standard quality rules to fit your needs.

Important!

A rule set must be activated to be used.

When a rule set is activated, it becomes available in the plug-in and it is possible to execute all of its quality rules to test the code quality of a source member.

The list of active rule sets is displayed in the  **Rule Sets** view. To access this view, either:

- right-click on a source member and select  **ARCAD CodeChecker** >  **Display rule sets**, or
- open the **Window** menu and select **Show View** > **Other...**, then open the **ARCAD CodeChecker** node and select  **Display rule sets**. Click **OK**.

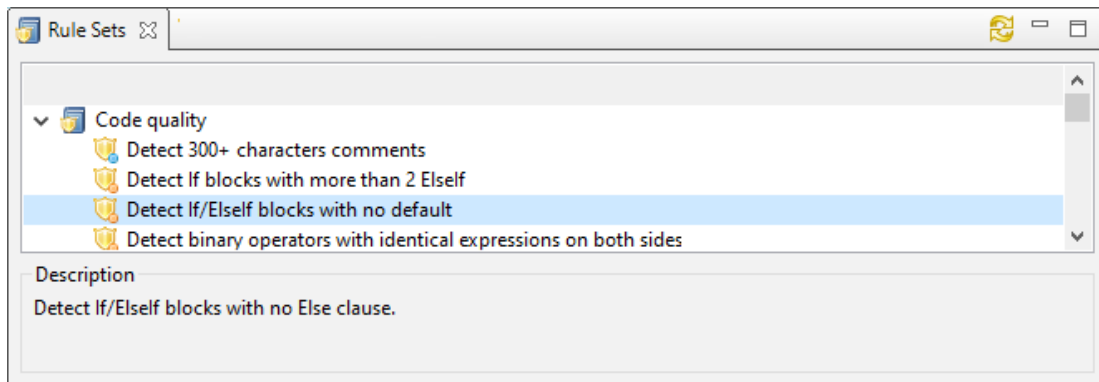




Figure 58: The Rule Sets view in RDi

The  **Rule Sets** view displays the list of active rule sets available, and gives more information about the rule sets and the quality rules they contain. In the rule sets, double-click on a quality rule to open its documentation in a your default web-browser.

Important!

Contact the person in charge of managing quality rules in the CodeChecker Studio if difficulties are encountered with the quality rules.

Rule set

Rule sets are identified by the  rule set icon and are used to organize quality rules in a logical way. Expand a rule set's node to display the quality rules contained in it.

Name

The names of the rule sets and quality rules are displayed:

- for rule sets, the name is followed by the rule set's ID between parentheses,
- for quality rules, the name is followed by the list of applicable languages between square brackets.




Warning!

Quality rules can only be executed on source members written in one of their applicable languages.

Failure level

The icons that identify each quality rule indicate the failure level given to each rule.

There are three hierarchical levels of failure in ARCAD CodeChecker:

-  FATAL, the most critical,
-  WARNING, and
-  NOTICE, the least critical.


The failure level given to a quality rule depends on the code quality that needs to be achieved.

Users with the *Rule Management* role in the CodeChecker Studio determine the failure level to give to each quality rule, and give the appropriate meaning to each failure level.

Description

Select a rule set or a quality rule to display its short description below the list of active rule sets. Double-click on a quality rule to open its documentation in a your default web-browser.

23.5 Executing quality rules

Executing quality rules on a source member means that the source code of that source member is analyzed and compared with the logical conditions defined in the quality rules. If the source code complies with the quality rules, the execution does not return errors in the  **Error List** view. If the source code does not comply with one or several quality rules, the execution returns errors with different failure levels depending on the failing quality rules.

Reference

For more information about the error list, refer to [Managing error lists on page 137](#).

There are several options to execute quality rules on a source member:

1. [Executing all the quality rules](#) contained in all the active rule sets.
2. [Selecting and execute quality rules](#) to only execute specific quality rules.
3. [Executing the last-used quality rules sets](#) to execute the last-used quality rules.

Important!

Quality rules should only be executed on a limited number of source member at a time for better performance. The number of concurrent executions can be set in the preferences.

Reference

For more information about the rule execution preferences, refer to [Setting rules execution preferences on page 132](#).


23.5.1 Executing all the quality rules


You can execute all of the quality rules to any selected source. Follow the subsequent steps to execute all the quality rules at once on a source member.

Step 1 From the **Remote Systems** view of the **Remote System Explorer** perspective, or within an i Project in the **i Projects** perspective, right-click on the source member to analyze.

Step 2 Select  **ARCAD CodeChecker** >  **Execute all quality rules**.

Result The execution starts in a batch job. The selected quality rules are loaded and applied one after the other to the selected source member.

The execution status is displayed in the  **Running Executions** view.

When the execution is complete, the results of the analysis are displayed in the  **Error List** view, which opens automatically.

23.5.2 Selecting and execute quality rules

Follow the subsequent steps to select specific quality rules to execute on a source member.

Step 1 From the **Remote Systems** view of the **Remote System Explorer** perspective, or within an i Project in the **i Projects** perspective, right-click on the source member to analyze.


Step 2 Select  **ARCAD CodeChecker** >  **Select and execute quality rules**.


Step 3 In the wizard, expand each rule set's node and check the boxes of the quality rules to execute.

By default, all the rule sets and quality rules are checked. Uncheck a rule set to automatically uncheck all its quality rules.

Click **Finish**.

Result The execution starts in a batch job. The selected quality rules are loaded and applied one after the other to the selected source member.

The execution status is displayed in the  **Running Executions** view.

When the execution is complete, the results of the analysis are displayed in the  **Error List** view, which opens automatically.


23.5.3 Executing the last-used quality rules sets


You can execute the last-used rule sets to any selected source. Follow the subsequent steps to re-execute the last-used rule sets to a source.

Step 1 From the **Remote Systems** view of the **Remote System Explorer** perspective, or within an i Project in the **i Projects** perspective, right-click on the source member to analyze.


Step 2 Select  **ARCAD CodeChecker** >  **Re-execute last-used rule set**.

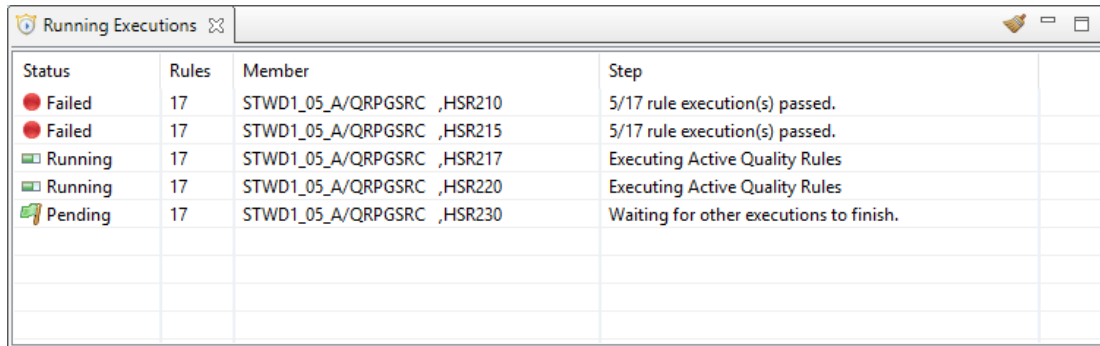
Result The execution starts in a batch job. The selected quality rules are loaded and applied one after the other to the selected source member.

The execution status is displayed in the  **Running Executions** view.

When the execution is complete, the results of the analysis are displayed in the  **Error List** view, which opens automatically.

23.6 Managing executions

The  **Running Executions** view displays all the executions launched. It enables you to get detailed information about the status of the execution (succeeded / failed, running, pending) and the execution duration and parse status.










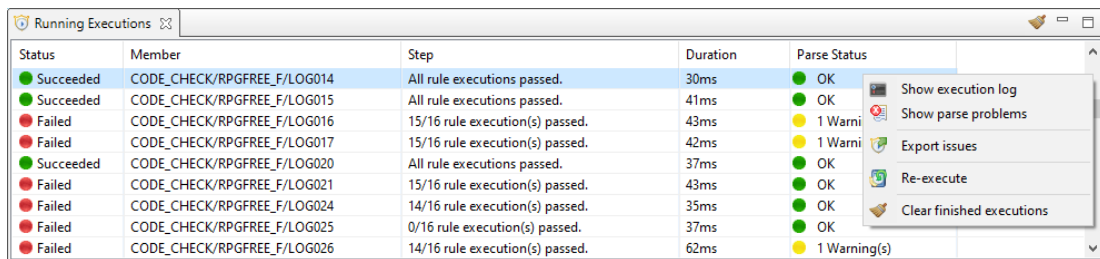
Status	Rules	Member	Step
 Failed	17	STWD1_05_A/QRPGSRC ,HSR210	5/17 rule execution(s) passed.
 Failed	17	STWD1_05_A/QRPGSRC ,HSR215	5/17 rule execution(s) passed.
 Running	17	STWD1_05_A/QRPGSRC ,HSR217	Executing Active Quality Rules
 Running	17	STWD1_05_A/QRPGSRC ,HSR220	Executing Active Quality Rules
 Pending	17	STWD1_05_A/QRPGSRC ,HSR230	Waiting for other executions to finish.

Figure 59: The Rules execution view in RDi

The execution logs and parse problems can be accessed from this view. Right-click on an execution result and choose the desired option the contextual menu:

-  **Show execution logs**
-  **Show parse problems**






















Status	Member	Step	Duration	Parse Status
 Succeeded	CODE_CHECK/RPGFREE_F/LOG014	All rule executions passed.	30ms	 OK
 Succeeded	CODE_CHECK/RPGFREE_F/LOG015	All rule executions passed.	41ms	 OK
 Failed	CODE_CHECK/RPGFREE_F/LOG016	15/16 rule execution(s) passed.	43ms	 1 Warni
 Failed	CODE_CHECK/RPGFREE_F/LOG017	15/16 rule execution(s) passed.	42ms	 1 Warni
 Succeeded	CODE_CHECK/RPGFREE_F/LOG020	All rule executions passed.	37ms	 OK
 Failed	CODE_CHECK/RPGFREE_F/LOG021	15/16 rule execution(s) passed.	43ms	 OK
 Failed	CODE_CHECK/RPGFREE_F/LOG024	14/16 rule execution(s) passed.	35ms	 OK
 Failed	CODE_CHECK/RPGFREE_F/LOG025	0/16 rule execution(s) passed.	37ms	 OK
 Failed	CODE_CHECK/RPGFREE_F/LOG026	14/16 rule execution(s) passed.	62ms	 1 Warning(s)

Figure 60: The Execution Logs view in RDi

To remove past executions, right-click in the view and choose  **Clear finished executions**.

23.6.1 Exporting issues

Follow the subsequent steps to export issues.

Step 1 Select the execution result the  **Running Executions** view. Multiple lines can be selected to aggregate the issues in a single export.

Step 2 Right-click on the selected execution result(s) and choose  **Export issues**.

Step 3 Select the export format (CSV, XML or JSoN) in the dialog. Click **OK**.

Step 4 Choose the name of the export file and the location where to save the .file. Click **Save**.

Result When the export is finished, a dialog is displayed to confirm the status of the export.

23.6.2 Re-executing past issues

Re-executing the rules of an execution allows to perform again the same quality rule checks on the same sources than the selected issue.

To re-execute past executions, select the execution result the **Running Executions** view. Multiple lines can be selected to re-execute several issues. Right-click on the selected execution result(s) and choose **Re-executing**. The execution results are refreshed in the view.

23.7 Managing error lists

When quality rules have been executed on a source member, the execution can return errors if the source code of the member did not comply with the quality rules. These errors are displayed and managed in the **Error List** view.

To access this view, either:

- right-click on a source member and select **ARCAD CodeChecker** > **Show latest execution results** to display the error list returned during the last execution, or
- execute quality rules on a source member.

Note

The **Error List** view opens automatically when the execution is complete.

Reference

For more information about executing quality rules, refer to [Executing quality rules on page 134](#).

23.7.1 Understanding an error list

The **Error List** view displays all the errors returned during the execution for each quality rule executed. If a quality rule is not displayed in this view, it means the source code complies with that quality rule.

ID	Message	Severity	Line	Location	Connection
Detect incorrect file name	File ord500o name must be uppercase.	0	9	CODE_CHECK/RPGFREE_PF(FP...	ARCAD_PGMR
Check file error handling	Missing INFSR declaration on ord500o	30	9	CODE_CHECK/RPGFREE_PF(FP...	ARCAD_PGMR
Detect unused files	Unused file ord500o	10	9	CODE_CHECK/RPGFREE_PF(FP...	ARCAD_PGMR
Detect incorrect file name	File custome1 name must be uppercase.	0	9	CODE_CHECK/RPGFREE_PF(FP...	ARCAD_PGMR
Check file error handling	Missing INFSR declaration on custome1	30	9	CODE_CHECK/RPGFREE_PF(FP...	ARCAD_PGMR
Detect unused files	Unused file custome1	10	9	CODE_CHECK/RPGFREE_PF(FP...	ARCAD_PGMR
Detect incorrect file name	File detord1 name must be uppercase.	0	9	CODE_CHECK/RPGFREE_PF(FP...	ARCAD_PGMR
Check file error handling	Missing INFSR declaration on detord1	30	9	CODE_CHECK/RPGFREE_PF(FP...	ARCAD_PGMR
Detect unused files	Unused file detord1	10	9	CODE_CHECK/RPGFREE_PF(FP...	ARCAD_PGMR

Figure 61: The Error List view in RDi

The **ID** column displays the quality rule's name. The icon indicating the failure level is displayed to the left of the **ID** column.

 **Reference**

For more information about failure levels, refer to [Failure level on page 134](#).

The **Message** column explains why the quality rule is failing. This message gives more information to help understand what is making the quality rule fail.


The **Severity** column displays a value depending on the failure level of the quality rule: *0* for NOTICE, *10* for WARNING, *30* for FATAL, and *40* for ERROR.


The **Line** column displays the line number of the source member where the error is located.

The **Location** column displays the source library and the source file of the member.

The **Connection** column displays the server on which the source member is located.

23.7.2 Correcting failures from an error list

Each failure represents an occurrence in the source code of the member where the condition defined in a quality rule was not respected. Follow the subsequent steps to correct the failures displayed in the  **Error List** view.

Step 1 In the  **Error List** view, double-click on the desired failure line.


Result If it was not already, the source member opens in an LPEX editor. The source member opens at the line number where the failure is located, and shows the failure in context. The failure is highlighted and is followed by the quality rule's name and the failure description, written in pink.

 **Note**

Some failures cannot be shown in context. For example, if a quality rule checks that there is at least one line of comment in the source code and the comment line does not exist, the program opens with the cursor on the first line and the failure is not highlighted.

Step 2 Modify the source code to correct the failure. The modifications will depend on the source code, on the failure, and on the condition defined in the quality rule.

Step 3 Save any changes made to the source code.

Step 4 Execute the quality rule on the source member again to refresh the list of failures in the  **Error List** view and to make sure that the failure is no longer detected in the list.

24 ARCAD CodeChecker for Jenkins

Chapter Summary

24.1 Prerequisites.....	139
24.2 Configuring the ARCAD CodeChecker for Jenkins plug-in.....	139
24.3 Setting up an ARCAD CodeChecker project.....	140
24.4 Executing campaigns in Jenkins.....	145

The ARCAD CodeChecker for Jenkins plug-in is intended for development team managers, or any person in charge of the evolution of an application's source code. It enables you to test the quality of a build and generate an analysis report for the build. With this report, you can quickly identify the files that do not comply with the quality rules executed during the code review process. The plug-in also enables you to stop the build process if a FATAL quality rule is failing.

24.1 Prerequisites

1. Jenkins must be installed. The plug-in is compatible with Jenkins \geq v2.190.1.



Reference

For more information about installing Jenkins, refer to the [Jenkins Installation Guide](#).

2. The ARCAD-Commons plug-in must be installed on Jenkins. This plug-in, common to many ARCAD products on Jenkins, only needs to be installed once, no matter how many ARCAD product plug-ins are installed on Jenkins. To install the ARCAD-Commons plug-in, click **Manage Jenkins** in the left menu of the dashboard, then click **Manage Plugins** and open the **Advanced** tab. In the **Upload Plugin** section, click **Choose File...** and navigate to the location where the file is stored. Select the *arcad-commons.hpi* file and click **Open**. Click **Upload**.

24.2 Configuring the ARCAD CodeChecker for Jenkins plug-in

Add the ARCAD CodeChecker for Jenkins plug-in to Jenkins and set the server connection details.

24.2.1 Adding the plug-in to Jenkins

Follow the subsequent steps to install an ARCAD plug-in on Jenkins.

Step 1 Open Jenkins in a navigator.

Step 2 To access the plug-in install page, click **Manage Jenkins** in the left menu of the dashboard, then click **Manage Plugins** and open the **Advanced** tab.

Step 3 In the **Upload Plugin** section, click **Choose File...** and navigate to the location where the installation file is stored. Select the *.hpi* file that corresponds to the product and click **Open**.

Step 4 Click **Upload**.

Step 5 To complete the installation, check **Restart Jenkins when installation is complete and no jobs are running**, and wait for Jenkins to restart.

Result The plug-in is installed and available for use.

24.2.2 Setting the connection details

Warning!

The following configuration is required.

Follow the subsequent steps to configure the ARCAD CodeChecker for Jenkins plug-in.

Step 1 To access the plug-in configuration page, click **Manage Jenkins** in the left menu of the dashboard, then click **Configure System** and find the **ARCAD CodeChecker** section.

Step 2 Define in the following fields:

URL

Define the URL address where the CodeChecker Server is located.

This URL is usually made up of a prefix, *http://* or *https://*, then followed by the server's host name (DNS name or IP address) and the server's port. By default, the CodeChecker Server uses the 5253 port number or the 52530 port number for the secured connection, but these numbers may have been changed during configuration.

Credentials

The login credentials are saved in the Jenkins keyring. Either select them from the list, or click the **Add** button to add new credentials to the keyring. The user and password must be those used to connect to the CodeChecker Server.

Step 3 Use the **Test connection** button to ensure that the connection is successful.

Step 4 To save any changes made, click **Save**.

Result The ARCAD CodeChecker for Jenkins plug-in is connected to the CodeChecker Server.

24.3 Setting up an ARCAD CodeChecker project

Setting up an ARCAD CodeChecker project in Jenkins corresponds to the creation of a code review campaign orchestrated from Jenkins.

24.3.1 Freestyle projects

Follow the subsequent steps to set up a freestyle ARCAD CodeChecker project in Jenkins.

Step 1 Create a Jenkins item. Click **New Item** in the menu.

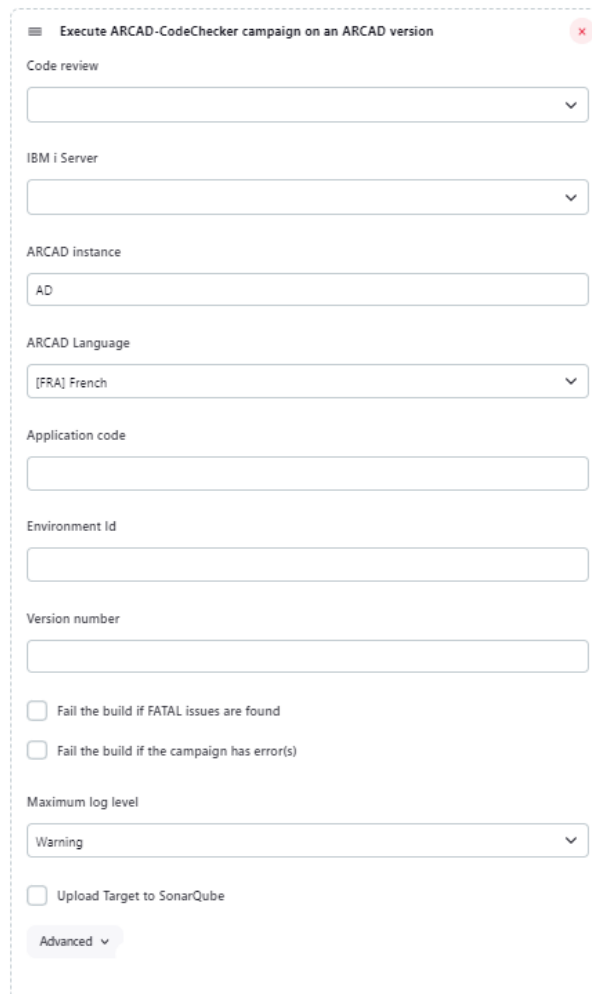
- Enter a name for the project.
- Choose **Freestyle project**.
- Click **OK**.

Step 2 In the **Add build step** drop-down list, select the appropriate option:

Select **Execute ARCAD CodeChecker predefined campaign** to create and execute a campaign from a code review and a target predefined in the CodeChecker Server.

Select **Execute ARCAD CodeChecker campaign on an ARCAD Version** to create and execute a campaign from a code review and an ARCAD version. This option will create a non-editable ARCAD target in the CodeChecker Server.

Select **Execute ARCAD CodeChecker campaign on source uploaded from Jenkins** to run a campaign from a local source folder. This job pulls the source code and uses ARCAD CodeChecker to analyze it during the build.



The screenshot shows a Jenkins configuration form for the build step "Execute ARCAD-CodeChecker campaign on an ARCAD version". The form includes the following fields and options:

- Code review:** A dropdown menu.
- IBM i Server:** A dropdown menu.
- ARCAD instance:** A text input field containing "AD".
- ARCAD Language:** A dropdown menu showing "[FRA] French".
- Application code:** A text input field.
- Environment Id:** A text input field.
- Version number:** A text input field.
- Fail the build if FATAL issues are found
- Fail the build if the campaign has error(s)
- Maximum log level:** A dropdown menu showing "Warning".
- Upload Target to SonarQube
- Advanced:** A dropdown menu.

Figure 62: The ARCAD CodeChecker code review build step in Jenkins

Step 3 Choose the campaign's details:

[If from a predefined campaign] Choose a **Code review** and a **Target** from the lists. All the code reviews and targets defined in CodeChecker Server are available from the lists.

[If from an ARCAD Version campaign] Choose a **Code review** and an **IBM i Server** from the lists. All the code reviews and IBM i Server connections defined in CodeChecker Server are available from the lists.

Specify the target **ARCAD instance** on the IBM i server and select the **ARCAD Language** (FRA or ENG). The selected language must be supported by the target ARCAD instance.

Specify the **Application code** of the Skipper application to be reviewed, the application's development **Environment ID** and the **Version number** of the version that must be tested.

[If from campaign on source uploaded from Jenkins] Choose a **Code review**, and an **IBM i Server** from the lists. All the code reviews and IBM i Server connections defined in CodeChecker Server are available from the lists.

Define a **Source folder** and set the code of the target using the **Target code** option. If the target code does not already exist, it is created with this value.

Step 4 Define the campaign execution options:

- *[Optional]* Tick the **Fail the build when Fatal issues are found** box to stop the Jenkins build process if a FATAL quality rule is failing.
- *[Optional]* Tick the **Fail the build if the campaign has error(s)** box to stop the Jenkins build process if an error occurs during the campaign execution.
- Choose a **Maximum log level** for the campaign (Error, Warning, Info, Debug).
- *[Optional]* Tick the **Upload Target to SonarQube** box to allow the upload for the target managed by this build step in SonarQube.
- *[If from campaign on source uploaded from Jenkins]* Set a root folder in the **Delta folder** option to have its content used to gather the reference issues and compare them to the issues found in the target source folder.
You can also define the size used to split the source package sent to the server by setting the **Upload spit size** option, which sets, in kilobytes, the upload size of the source files. By default, the value is set to 5120 Kb.

In the **Advanced** options:

- *[Optional]* Tick the **Clear existing issues before execution** box to clear the target issues before the step is executed.
- *[Optional]* Tick the **Generate Sonar Project** box and enter the **Project location** to set the directory in which the campaign results will be downloaded as a Sonar project. The campaign execution task will create a `.sonar-issues` JSON file at the root of the job's workspace and download the target's source files in the project location directory.

Click **Save**.

Result The code review campaign is set up in Jenkins.

24.3.2 Pipeline projects

Follow the subsequent steps to set up a pipeline ARCAD CodeChecker project in Jenkins.

Step 1 Create a Jenkins item. Click **New Item** in the menu.

- Enter a name for the project.
- Choose **Pipeline project**.
- Click **OK**.

Step 2 Select the definition of the pipeline. Use pipeline script.

Step 3 Generate the pipeline script. Click the **Pipeline Syntax** link to open the Snippet Generator page.

Step 4 In the **Sample Step** drop-down list, select the appropriate option:

Select **Execute ARCAD CodeChecker predefined campaign** to create and execute a campaign from a code review and a target predefined in the CodeChecker Server.

Select **Execute ARCAD CodeChecker campaign on an ARCAD Version** to create and execute a campaign from a code review and an ARCAD version. This option will create a non-editable ARCAD target in the CodeChecker Server.

Select **Execute ARCAD CodeChecker campaign on source uploaded from Jenkins** to run a campaign from a local source folder. This job pulls the source code and uses ARCAD CodeChecker to analyze it during the build.

Step 5 Choose the campaign's details:

[If from a predefined campaign] Choose a **Code review** and a **Target** from the lists. All the code reviews and targets defined in CodeChecker Server are available from the lists.

[If from an ARCAD Version campaign] Choose a **Code review** and an **IBM i Server** from the lists. All the code reviews and IBM i Server connections defined in CodeChecker Server are available from the lists.

Specify the target **ARCAD instance** on the IBM i server and select the **ARCAD Language** (FRA or ENG). The selected language must be supported by the target ARCAD instance.

Specify the **Application code** of the Skipper application to be reviewed, the application's development **Environment ID** and the **Version number** of the version that must be tested.

[If from campaign on source uploaded from Jenkins] Choose a **Code review**, and an **IBM i Server** from the lists. All the code reviews and IBM i Server connections defined in CodeChecker Server are available from the lists.

Define a **Source folder** and set the code of the target using the **Target code** option. If the target code does not already exist, it is created with this value.

Step 6 Define the campaign execution options:

- *[Optional]* Tick the **Fail the build when Fatal issues are found** box to stop the Jenkins build process if a FATAL quality rule is failing.
- *[Optional]* Tick the **Fail the build if the campaign has error(s)** box to stop the Jenkins build process if an error occurs during the campaign execution.
- Choose a **Maximum log level** for the campaign (Error, Warning, Info, Debug).
- *[Optional]* Tick the **Upload Target to SonarQube** box to allow the upload for the target managed by this build step in SonarQube.
- *[If from campaign on source uploaded from Jenkins]* Set a root folder in the **Delta folder** option to have its content used to gather the reference issues and compare them to the issues found in the target source folder.
You can also define the size used to split the source package sent to the server by setting the **Upload spit size** option, which sets, in kilobytes, the upload size of the source files. By default, the value is set to 5120 Kb.

In the **Advanced** options:

- [Optional] Tick the **Clear existing issues before execution** box to clear the target issues before the step is executed.
- [Optional] Tick the **Generate Sonar Project** box and enter the **Project location** to set the directory in which the campaign results will be downloaded as a Sonar project. The campaign execution task will create a `.sonar-issues` JSON file at the root of the job's workspace and download the target's source files in the project location directory.

Step 7 Click **Generate Pipeline Script**. The snippet for the script is generated in the box.

Step 8 Copy the snippet and paste it in the pipeline script in the project's configuration page.

The syntax of the script should be:

Execute CodeChecker predefined campaign

```
1 | node {
2 |     codeCheckerExecute codeReview: [code: 'RPG1'],
3 |     campaignLogLevel: [maxLevel: 'WARNING'],
4 |     target: [code: 'TRGT1'],
5 |     generateSonarReport: [sourceDownloadLocation: 'cc-sources'],
6 |     failOnError: true,
7 | }
```

Execute CodeChecker campaign on an ARCAD Version

```
1 | node {
2 |     arcadCodeCheckerExecute applicationCode: 'ARCAD_DEMO',
3 |     arcadInstance: 'AD',
4 |     arcadLanguage: [code: 'ENG'],
5 |     campaignLogLevel: [maxLevel: 'WARNING'],
6 |     codeReview: [code: 'CR1'],
7 |     environmentId: 'ABCD',
8 |     ibmi: [code: 'ARCAD123'],
9 |     versionNumber: '1.2.3',
10 |     resetTarget: true
11 | }
```

Execute ARCAD CodeChecker campaign on source uploaded from Jenkins

```
1 | node {
2 |     codeCheckerExternalExecution
3 |     campaignLogLevel: [maxLevel: 'WARNING'],
4 |     codeReview: [code: 'QA'],
5 |     deltaFolder: '',
6 |     description: 'This is a description',
7 |     ibmi: [code: 'HOM01'],
8 |     sourceFolder: 'BBS400/src',
9 |     splitSize: 5120,
10 |     targetCode: 'JENKINS'2.3',
11 | }
```

Click **Save**.

Result The ARCAD CodeChecker test campaign is set up in Jenkins.

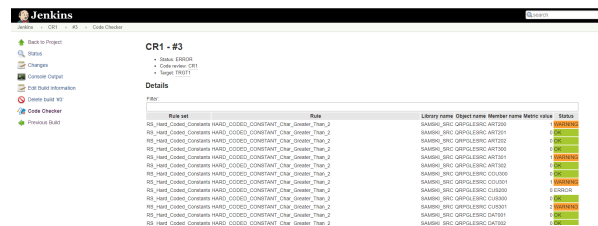
24.4 Executing campaigns in Jenkins

Execute the code review campaign and see the results with Jenkins.

Step 1 To launch the code review campaign from Jenkins, click **Build Now**.

Step 2 When the build is finished, click the link to open the build details and choose **Code Checker** in the menu.

Result The results of the code review campaign are displayed. A dynamic search bar allows you to navigate the results.



Rule set	Rule	Library name	Object name	Message name	Method name	Severity	Task
RS_Hard_CodeCheck_Constraints_WA0_CODED_CONSTANT_Clar_Swains_Team_2		SAWINS_SRC_GOPHLESPIC_ART200				PASS	
RS_Hard_CodeCheck_Constraints_WA0_CODED_CONSTANT_Clar_Swains_Team_2		SAWINS_SRC_GOPHLESPIC_ART201				PASS	
RS_Hard_CodeCheck_Constraints_WA0_CODED_CONSTANT_Clar_Swains_Team_2		SAWINS_SRC_GOPHLESPIC_ART202				PASS	
RS_Hard_CodeCheck_Constraints_WA0_CODED_CONSTANT_Clar_Swains_Team_2		SAWINS_SRC_GOPHLESPIC_ART203				PASS	
RS_Hard_CodeCheck_Constraints_WA0_CODED_CONSTANT_Clar_Swains_Team_2		SAWINS_SRC_GOPHLESPIC_ART204				PASS	
RS_Hard_CodeCheck_Constraints_WA0_CODED_CONSTANT_Clar_Swains_Team_2		SAWINS_SRC_GOPHLESPIC_ART205				PASS	
RS_Hard_CodeCheck_Constraints_WA0_CODED_CONSTANT_Clar_Swains_Team_2		SAWINS_SRC_GOPHLESPIC_ART206				PASS	
RS_Hard_CodeCheck_Constraints_WA0_CODED_CONSTANT_Clar_Swains_Team_2		SAWINS_SRC_GOPHLESPIC_ART207				PASS	
RS_Hard_CodeCheck_Constraints_WA0_CODED_CONSTANT_Clar_Swains_Team_2		SAWINS_SRC_GOPHLESPIC_ART208				PASS	
RS_Hard_CodeCheck_Constraints_WA0_CODED_CONSTANT_Clar_Swains_Team_2		SAWINS_SRC_GOPHLESPIC_ART209				PASS	
RS_Hard_CodeCheck_Constraints_WA0_CODED_CONSTANT_Clar_Swains_Team_2		SAWINS_SRC_GOPHLESPIC_ART210				PASS	
RS_Hard_CodeCheck_Constraints_WA0_CODED_CONSTANT_Clar_Swains_Team_2		SAWINS_SRC_GOPHLESPIC_ART211				PASS	
RS_Hard_CodeCheck_Constraints_WA0_CODED_CONSTANT_Clar_Swains_Team_2		SAWINS_SRC_GOPHLESPIC_ART212				PASS	
RS_Hard_CodeCheck_Constraints_WA0_CODED_CONSTANT_Clar_Swains_Team_2		SAWINS_SRC_GOPHLESPIC_ART213				PASS	
RS_Hard_CodeCheck_Constraints_WA0_CODED_CONSTANT_Clar_Swains_Team_2		SAWINS_SRC_GOPHLESPIC_ART214				PASS	

Figure 63: The campaign results in Jenkins

25 ARCAD CodeChecker for SonarQube

Chapter Summary

25.1 Enabling results upload for a target	146
25.2 Accessing the results in SonarQube.....	147

The ARCAD CodeChecker solution for SonarQube integrates the results of code quality campaign in SonarQube, the open-source platform for continuous inspection of code quality. You can review issues raised in a campaign with the functionalities offered by the Sonar interface (code review, statistics, etc.). Like in the CodeChecker Studio, rule documentation is available for each issue in Sonar.

The solution is composed of:

- The ARCAD CodeChecker plug-in for SonarQube, to be installed on the SonarQube server. The plug-in is a SonarQube add-on that allows you to manage your source code quality with SonarQube.
- The Sonar Scanner, supplied by Sonar and fully managed on the CodeChecker Server side. The scanner is a Sonar component that ensures the communication of the CodeChecker Server with SonarQube.

The plug-in interfaces ARCAD CodeChecker with SonarQube. The role of the plug-in is as follows:

- At Sonar server start, the plug-in loads all the rules in a **Quality Profile** named **CodeChecker**, and there is one Quality Profile per programming language. Each rule has a definition that includes the rule's information and its documentation.
- When results are sent from ARCAD CodeChecker to SonarQube, the plug-in reads the JSON report generated by ARCAD CodeChecker. This file contains the list of issues found during a campaign. The issues are then loaded in SonarQube.

The Sonar Scanner's role is to send the campaigns results from ARCAD CodeChecker to SonarQube. The scanner is installed and managed via the CodeChecker Server in the Preferences of the CodeChecker Studio.


Reference

For more information about the installation instruction for the plug-in and the Sonar Scanner, refer to the [ARCAD CodeChecker Installation Guide](#).

For more information about the connection between ARCAD CodeChecker and SonarQube, refer to the [Sonar Integration preferences on page 36](#).

25.1 Enabling results upload for a target

The results can be automatically uploaded from ARCAD CodeChecker to SonarQube via  [Targets](#).

To make the upload of results possible, enable the **Upload to SonarQube** option in the **Target Definition** section of the  Target editor.

Note


The **Upload to SonarQube** option is automatically disabled if no preferences are set for SonarQube.


Targets for which the upload of results is enabled display the  SonarQube icon in the  **Targets** list.


The upload of results is performed at the end of each campaign execution for a target on which the **Upload to SonarQube** option has been enabled.

25.2 Accessing the results in SonarQube...

... from the **Target list**

You can access results in SonarQube directly from the  **Targets** search view in the CodeChecker Studio.


To do so, right-click on a target and click on the  **Open issues in SonarQube** option.





You can also click the  SonarQube icon corresponding to the target.

Important!

To access results from a target, the **Upload to SonarQube** option has to be enabled in the **Target Definition** section of the target options.

... from the **Campaign results**

You can access results in SonarQube directly from the **Campaign Definition** section of the  **Campaign results** view in the CodeChecker Studio.

To do so, go to the  **Campaigns** search view and either double-click on a campaign or right-click on a campaign and select  **Open Campaign Results** to open the  **Campaign results** view, then click on the  **Open issues in SonarQube** option.

Important!

To access results from the  **Campaign results**, the **Upload to SonarQube** option has to be enabled in the **Target Definition** section of the target options.



APPENDICES

Dashboard metrics

The REST API web service allows external tools to retrieve the measures for a given target. Use the service to exploit the measures recorded during the code review campaigns in other tools, typically for dashboard purposes.

Reference

For more information about measures, refer to [Measures on page 89](#).

For more information about the use of measure data in ARCAD Dashboard, refer to the [ARCAD Dashboard documentation](#).

The web service URL is `/measures/{targetId}`. The web service gives the result in JSON format.

Attribute

targetId: the ID of the target for which measures will be retrieved.

Parameters

groupby: define how the measures should be regrouped in the resulting JSON. Possible values are:

- **source**: The answer is an array of JSON elements, where values are regrouped by source member. Each element contains a sub-array of measures regrouped by metric.
- **metric**: The answer is an array of JSON elements, where values are regrouped by metric. Each element contains a sub-array of measures regrouped by source member.
- **raw**: this is the default value when **groupby** is not specified. The JSON is a simple array where each element is a measure. No grouping is done.
- **last**: only the measures from last campaign executed for the target are returned.

Example

Here are some examples of web service calls:

To get all the measures for the Target #1, grouped by metric

```
http://codechecker-  
server:5253/measures/1?groupby=metric
```

To get all the measures for the Target #8, grouped by source

```
http://codechecker-  
server:5253/measures/8?groupby=source
```

To get the latest measures for the Target #1, grouped by source

```
http://codechecker-  
server:5253/measures/1?groupby=source&last
```



To get the latest measures for the Target #2

`http://codechecker-server:5253/measures/2?last`

Related web services

The web service below can be called to retrieve information about items related to the measures:

To get the information about the Campaign #{id}

`http://codechecker-server:5253/data/campaign/ {id}`

To get the information about the Target #{id}

`http://codechecker-server:5253/data/target/ {id}`

To get the information about the Metric #{id}

`http://codechecker-server:5253/data/metric/ {id}`

Glossary

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A

Activated

A state applied to rule sets and quality rules to make them usable.

Rule sets must be activated to be used in code reviews or in the ARCAD CodeChecker plug-in for RDi.

Quality rules must be activated to be executed and measure code quality, and to be added to rule sets.

Activation

A user role that enables users to activate or deactivate rule sets and to activate quality rules.

Analysis report

A report available in the ARCAD CodeChecker plug-in for Jenkins.

When a build that was launched using Jenkins is finished, ARCAD CodeChecker generates an analysis report that displays the results of the code review process.

Applicable languages

The languages that can be analyzed by ARCAD CodeChecker. The languages recognized by ARCAD CodeChecker are:

- CL
- COBOL
- RPG (RPG III)
- RPGLE (RPG IV)

The applicable languages are selected at the metric level and define the languages that can be analyzed using the metric. If a certain language is not selected for a metric, the quality rules using this metric cannot be executed on applications written in that language.

Application

A term that refers to the source code analyzed during a code review process.

ARCAD CodeChecker

A licensed software application that automates the tasks of analyzing code quality, detecting complexity hotspots in your code, and ensuring a consistent level of quality throughout your application.

ARCAD CodeChecker plug-in for Jenkins

A plug-in designed for Jenkins, an open-source tool for continuous integration.

The ARCAD CodeChecker for Jenkins plug-in is intended for development team managers, or any person in charge of the evolution of an application's source code. It enables you to test the quality of a build and generate an analysis report for the build. With this report, you can quickly identify the files that do not comply with the quality rules executed during the code review process. The

plug-in also enables you to stop the build process if a FATAL quality rule is failing.

ARCAD CodeChecker plug-in for RDi

The ARCAD CodeChecker plug-in for RDi is designed for the RDi integrated development environment (IDE).

The plug-in is designed for developers working in the RDi IDE. It enables developers to test the quality of the source code they are working on by executing active rule sets or specific quality rules, and know instantly if the source code does not comply with the standards set for the development team. With the plug-in, developers can easily know which quality rules are failing and identify which part of their source code is causing the failure. The portions of code causing the failures are clearly identified, making it easy for developers to modify their source code to comply with the quality standards.

ARCAD CodeChecker plug-in for SonarQube

The ARCAD CodeChecker solution for SonarQube integrates the results of code quality campaign in SonarQube, the open-source platform for continuous inspection of code quality. You can review issues raised in a campaign with the functionalities offered by the Sonar interface (code review, statistics, etc.). Like in the CodeChecker Studio, rule documentation is available for each issue in Sonar.

The solution is composed of:

- The ARCAD CodeChecker plug-in for SonarQube, to be installed on the SonarQube server. The plug-in is a SonarQube add-on that allows you to manage your source code quality with SonarQube.
- The Sonar Scanner, supplied by Sonar and fully managed on the CodeChecker Server side. The scanner is a Sonar component that ensures the communication of the CodeChecker Server with SonarQube.

ARCAD CodeChecker navigator

The primary view of the CodeChecker Studio, which gives access to all the elements required to set up and carry out the code review process.

B

C

Campaign

The execution of a code review on a target. Campaigns can be scheduled or manually created and launched.

Campaigns are created by associating a code review with a target and are used to keep a record of the executions over time. When a campaign is launched, the rule sets defined in the selected code review are executed on the associated target. When a campaign is completed, the campaign status indicates if the campaign succeeded or failed.

Campaign results

The results available when a campaign is completed, to get information about the code review process and know which quality rules are failing and, if so, which source members are causing the failure.

Campaign status

The status of a completed campaign that indicates if the campaign succeeded or failed.

After the code review process is finished, the campaign status is either changed to OK or to one of the failure levels: NOTICE, WARNING, or FATAL.

- If the campaign status is OK, it means that none of the quality rules executed failed during the code review process.
- If the campaign status is one of the failure levels, it means that at least one of the quality rules executed failed during the code review process. The failure level given to the quality rule determines the campaign status. If multiple quality rules with multiple failure levels failed during the code review process, the most critical failure level is used to determine the campaign status.

CCSID

Coded Character Set Identifier. Specific to IBM i. A 16-bit number that includes a specific set of encoding scheme identifiers, character set identifiers, code page identifiers, and other information that uniquely identifies the coded graphic-character representation.

Check

An execution of a quality rule on one source member during a code review process.

CodeChecker Server

The CodeChecker Server stores all the elements required to set up and carry out a code review process, and gives access to the predefined metric models and other entities that are delivered with ARCAD CodeChecker. The server is also in charge of carrying out the code review process.

CodeChecker Studio

The CodeChecker Studio is an Eclipse-based application that connects to the CodeChecker Server. It enables you to create and manage all the entities required to set up and carry out the code review process. The CodeChecker Studio is available as a stand-alone Rich Client Platform (RCP) and as an RDi perspective.

Code review

The context of a code quality analysis, which defines the rule sets that must be used during the code review process.

Code reviews enable users to specify an ensemble of coherent quality rules that need to be executed on a defined target application to check the code quality. Code reviews are associated with targets to create campaigns. When a campaign is launched, the rule sets defined in the code review are executed on the associated target.

Code Review Management

A user role that enables users to access, create and manage all the entities required to carry out the code review process: code reviews, targets, campaigns and schedules.

Code review process

A term that refers to the act of analyzing the code quality of an application. The code review process is carried out when a campaign is launched.

Campaigns are created using code reviews and targets to define the context of the code analysis. When a campaign is launched, the rule sets defined in the code review are executed on the associated target. Each of the quality rules contained in the rule sets are executed on each of the source members specified in the target. If a source member complies with the logical condition

defined in a quality rule, then the rule succeeds. If the source member does not comply, then the rule fails.

Configuration Management

A user role that enables users to access, create and manage all the entities required to set up the code review process: rule sets, quality rules, metrics, metric models and validation expressions.

D

E

Error list

A list available in the ARCAD CodeChecker plug-in for RDi.

When quality rules are executed on source code in RDi, it opens a new view containing the list of all the failures detected during the analysis.

F

Failure level

The level given to a quality rule to indicate whether it is critical or not for the rule to succeed.

There are three hierarchical levels of failure in ARCAD CodeChecker:

- 🚫 FATAL, the most critical,
- ⚠️ WARNING, and
- 📌 NOTICE, the least critical.

The failure level given to a quality rule depends on the code quality that needs to be achieved.

Users with the *Rule Management* role in the CodeChecker Studio determine the failure level to give to each quality rule, and give the appropriate meaning to each failure level.

FATAL

One of the three failure levels given to quality rules. The FATAL failure level is the most critical one.

G

Groovy

A scripting language used in ARCAD CodeChecker to create customized metric models and validation expressions.

H

I

IBM i

IBM i is an operating system that runs on IBM Power Systems and on IBM PureSystems. This integrated operating environment brings together robust architecture, high security and business resilience.

IFS

Integrated File System. The file system of the IBM i.

Issues

Issues are the individual results of the quality evaluation of the source code. They are created when a campaign is launched and are then stored on the server independently from the campaign that uncovered them. The issues' delta is calculated when the campaign is re-executed.

J

Jenkins

An open-source tool that allows for continuous integration. Jenkins is installed on a server and enables developers to reliably build, test and deploy their applications.

K

L

Languages

See [Applicable languages](#).

M

Metric

A numeric value deduced from the analysis of an application's source code.

Metrics are based on metric models. A metric model is a script that defines the actions to carry out and their required properties. The metric created from a metric model must be instantiated to be used in a quality rule: this means values for each of the properties are defined to conform to the specific quality rule's context.

Metric model

A set of instructions used to create metrics and analyze an application's source code.

Metric models define the actions to carry out to retrieve a value during the code analysis. Metric models use properties. When the metric using a metric model is instantiated for a quality rule, values for each of the properties are defined to conform to the specific rule's context.

N

NOTICE

One of the three failure levels given to quality rules. The NOTICE failure level is the least critical one.

O

P

Parameter

A type of variable used in validation expressions.

The formula defined in a validation expression can use parameters, which are elements that can be replaced by any value. Values for each of the parameters are defined in the context of a quality rule.

Plug-in

An add-on for a program that adds functionality to it.

ARCAD CodeChecker is available as a plug-in for integrated modules: RDi, SonarQube and Jenkins. These plug-ins make it possible to use specific ARCAD CodeChecker features outside of the CodeChecker Studio.

Property

A type of variable used in metric models.

The instructions defined in a metric model can use properties, which are elements that can be replaced by any value. Values for each of the properties are defined when a metric is instantiated in the context of a quality rule.

Q

Quality rule

A conditional expression used to determine an application's code quality.

Quality rules must be activated to be executed and to measure code quality. A simple quality rule is built around a metric and a validation expression. During a code review process, the metric is compared with the logical condition checked by the validation expression:

- If the metric complies with the condition of the validation expression, the quality rule succeeds.
- If the metric does not comply with the condition, the quality rule fails.

R

RDi

An integrated development environment (IDE) built on the Eclipse platform and owned by IBM. RDi is designed for creating and maintaining applications on IBM i systems.

Role

An access right assigned to a user profile to allow or deny access to specific features of the CodeChecker Studio.

There are 5 user roles available in ARCAD CodeChecker:

- [Server Administration](#),
- [Configuration Management](#),
- [Activation](#),
- [User Management](#),
- [Code Review Management](#).

Rule set

An entity used to organize quality rules in a logical way: by language, by theme, and so on. Users decide how to manage rule sets and categorize quality rules.

S

Schedule

An entity used to automatically create and launch campaigns at a specified date and time or at a specified interval of time. Schedules can be defined to create one campaign at a later date, or to create multiple campaigns over time, following a recurring schedule.

Scheduled campaign

A campaign that was created by a schedule. Scheduled campaigns are not created manually, but are automatically created and launched by ARCAD CodeChecker following a defined schedule, whether recurring or not.

Script

A script is the content of a metric model. It defines how the metric computes its value from the source and properties.

See [Metric model on page 155](#)

Server

See [CodeChecker Server](#).

Server Administration

A user role that enables users to configure the CodeChecker Server.

SonarQube

An open-source platform that allows for automatic reviews with static analysis of code to detect bugs, code smells and security vulnerabilities.

Source member

A set of statements written in a computer language and used to build IBM i programs and ARCAD applications. Source members contain the source code to be submitted for a compilation process.

Studio

See [CodeChecker Studio](#).

T

Target

An entity that enables users to identify the application and the specific source members to analyze during a code review process.

Targets are associated with code reviews to create campaigns. When a campaign is launched, the rule sets defined in the code review are executed on the associated target.

There are three types of targets:

- IBM i targets, which enable users to connect to IBM i and to specify the libraries, objects and members to include in the code review process.
- ARCAD targets, which enable users to connect to an ARCAD server and to specify the application, environment and version to include in the code review process.
- Git targets, which enable users to clone a Git repository and include its content to the code review process.

U

User

Password protected accounts that define the access rights for the different members of your team that use ARCAD CodeChecker. Each user account has a unique login.

User Management

A user role that enables users to access, create and manage user profiles.

V

Validation expression

A logical condition used in the context of a quality rule.

During the code review process, the validation expression is compared with the metric also defined in the quality rule to return a boolean value (true or false). The result of this comparison determines if the quality rule succeeds or fails.

ARCAD CodeChecker comes with a set of standard validation expressions that are designed to cover most needs. It is also possible to create specific validation expressions if needed.

W

WARNING

One of the three failure levels given to quality rules.

X

Y

Z
