Instructions

Fabasoft app.ducx extension for Visual Studio Code

Valid from 19.09.2024

For internal use



Copyright © Fabasoft R&D GmbH, Linz, Austria, 2025.

All rights reserved. All hardware and software names used are registered   
trade names and/or registered trademarks of the respective manufacturers.

No rights to our software or our professional services, or results of our   
professional services, or other protected rights can be based on the handing   
over and presentation of these documents.

Contents

[Fabasoft app.ducx extension for Visual Studio Code 4](#_Toc199249930)

[Install an Extension 4](#_Toc199249931)

[Extension Features 4](#_Toc199249932)

[Requirements 4](#_Toc199249933)

[Create app.ducx Projects 4](#_Toc199249934)

[Add Source Files 5](#_Toc199249935)

[Building and Running 5](#_Toc199249936)

[Gradle Tasks 6](#_Toc199249937)

[Execute Unit Tests 9](#_Toc199249938)

[Code Coverage 10](#_Toc199249939)

[Evaluate Expression 10](#_Toc199249940)

[Feedback 10](#_Toc199249941)

[Support for Large Workspaces 11](#_Toc199249942)

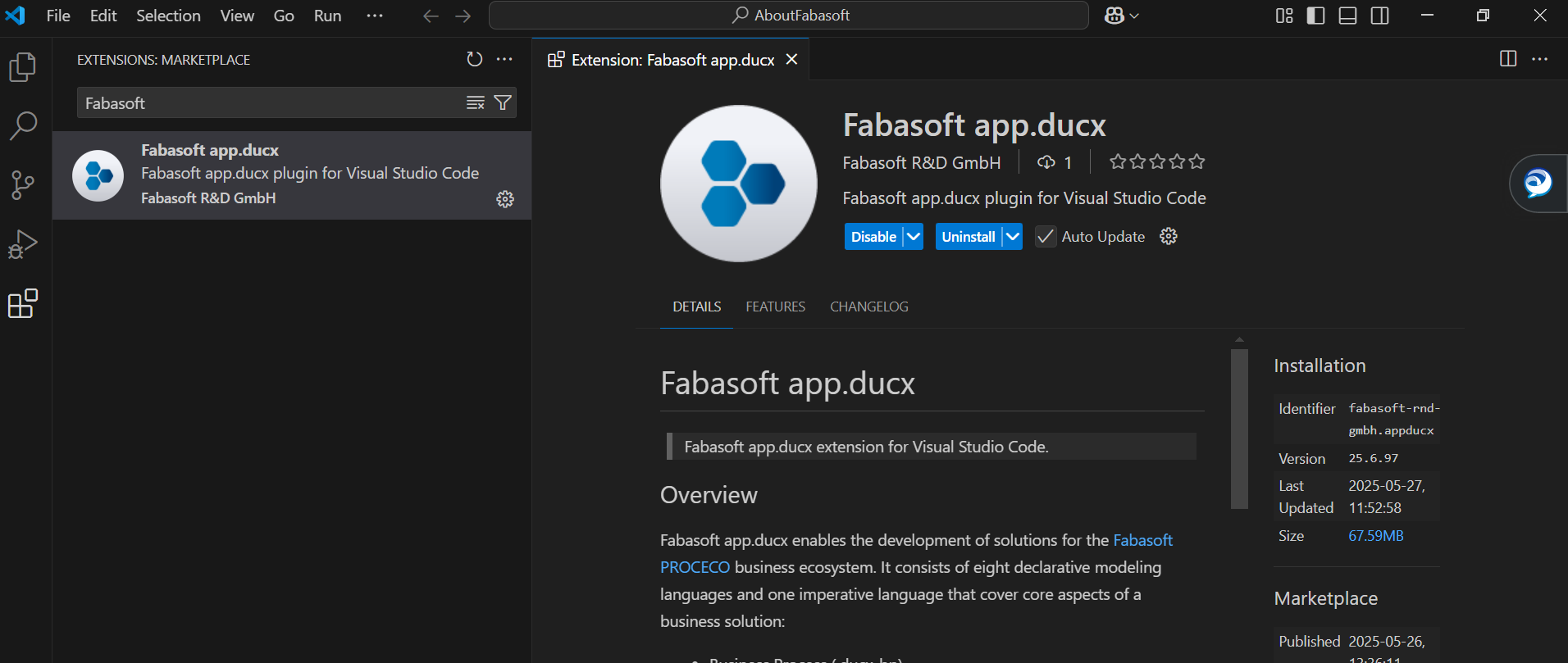
# Fabasoft app.ducx extension for Visual Studio Code

The Fabasoft app.ducx extension is now available for Visual Studio Code! Enhance your development experience by exploring the features of [Visual Studio Code](https://code.visualstudio.com/docs/editor/codebasics) alongside our extension.

## Install an Extension

The extension is now available on Visual Studio Code Marketplace and you can easily download it via VSCode application. For installation follow these steps:

1. In the extensions panel search for Fabasoft app.ducx
2. Click the “install” button



## Extension Features

The app.ducx Extension for Visual Studio Code offers you a wide range of features that enhances your development experience. For Fabasoft app.ducx extension features overview visit the [link](https://at.cloud.fabasoft.com/folio/fscasp/content/bin/fscvext.dll?bx=COO.6505.100.24.2290258&venv_view=COO.1.1001.1.38478).

Refer to the official Visual Studio Code documentation for detailed information. It offers a thorough guide to help you understand and make the most of these functionalities.

## Requirements

In order to use Fabasoft app.ducx extension for Visual Studio Code, you need to have at least java 21 installed and the java command as well as sh in your PATH.

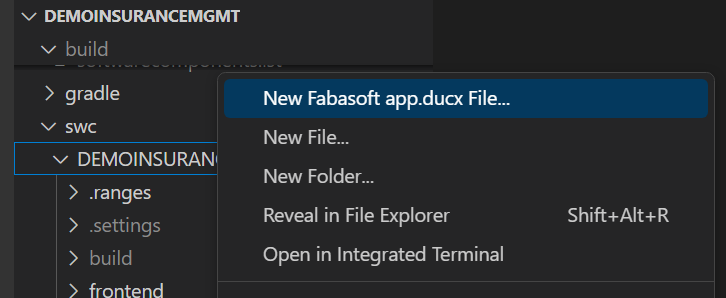
## Create app.ducx Projects

Building and creating app.ducx projects is/will be handled outside of VS Code.

There is a template for a solution using the gradle plugin (<https://enggit.fabafsc.fabagl.fabasoft.com/fsc/ducx/solutions-template>). For other use cases, you may still want to use eclipse to set up the project.

### Add Source Files

To add app.ducx source files with a template model block, right-click on the project folder and select “New Fabasoft app.ducx File…” from the context menu. Then, choose the preferred file extension from the dropdown.



To ensure proper setup, verify that the environment variable DUCX\_COPYRIGHT is defined and referenced in the .ducxproject file. Note that without a .ducxproject file, you won't be able to create a new Fabasoft app.ducx file with a template. However, you can still insert a model block snippet if you’re creating an empty file manually.

## Building and Running

To build and run container solutions, it's recommended to install the [Gradle extension](https://marketplace.visualstudio.com/items?itemName=richardwillis.vscode-gradle-extension-pack). This extension allows you to execute various Gradle commands from a new tab in Visual Studio Code.

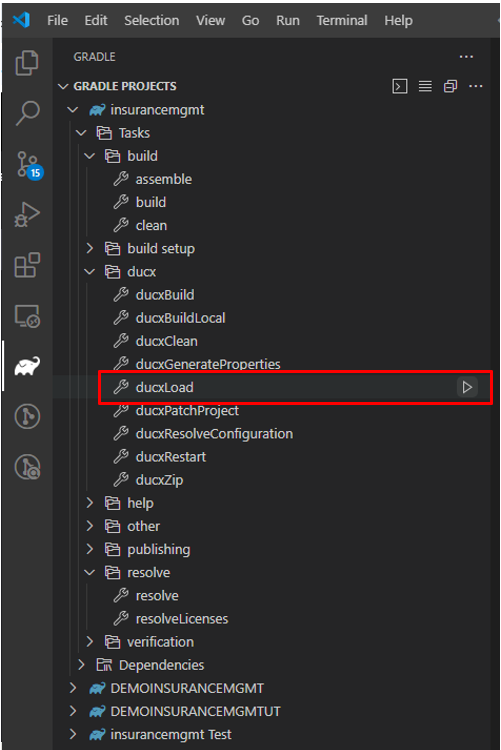
There should not be any errors present in the workspace before uploading a project. To successfully upload your project, run the following commands:

1. ./gradlew ducxBuildLocal
2. ./gradlew ducxLoad ducxRestart

Ensure that the .ducxws file from the root directory of your solution is correctly configured before uploading the project.

**Note:** ducxRestart is only necessary if using a container backend.

If you prefer using the UI, open the Gradle view and run tasks by clicking the run icon, as shown in the screenshot.



### Gradle Tasks

For Fabasoft app.ducx custom gradle tasks are available.

|  |  |
| --- | --- |
| Gradle Task | Description |
| ducxBuild | Builds the solution or a single project. |
| ducxBuildLocal | Builds the solution or a single project with the possibility to create new objects. Updates .ranges files and language files. |
| ducxLoad | Uploads the solution. |
| ducxRestart | Restarts all services. |
| ducxClean | Removes the result of ducxBuild/ducxBuildLocal. |
| ducxCleanLanguages | Cleans language files, converts the language files to CSV format. |
| ducxAddRange | Adds a range to an existing or new range file. |
| ducxAddTranslation | Adds other languages to one or more language files, converts the language files to CSV format. |
| ducxGenerate | Generates required libraries for language integrations |
| ducxGenerateOpenAPI | Generates OpenAPI clients and servers based on a specification. |
| ducxPatchProject | Adds java settings to a project. |
| ducxStartCoverage | Starts a coverage session |
| ducxStopCoverage | Stops a coverage session |
| ducxTrace | Starts a trace session. |
| ducxRunTests | Executes the specified app.ducx unit tests on the server. |

ducxAddRange

Add a new range.

The option –-name specifies the base name of the range file for the new range. If omitted, the current range file is used (see app.ducx documentation).

The option *--*create forces the creation of a new range file.

The options --ducxUsername and –ducxPassword or the properties ducxUsername and ducxPassword are used as credentials for the range service. Properties can be specified on the gradle command line with -P or in a gradle.properties file in a user specific location (see gradle documentation).

If username or password are not set, the task asks for it interactively. Note: It is not possible to hide the input in the gradle console for the password prompt.

The default range service is the Fabasoft Cloud, so the username is the email address of a regular user and the password is an “Access for Application” password for the Application “App Development”.

Alternatively it is possible to create a .range.ducxws file with the range service and the credentials.

ducxAddTranslation

Adds other languages to the language files of the project to create CSVs for translation.

This task uses the arguments target and source to specify which languages should be used.

If an argument is not set, all languages of the project are used.

The command to get the translation files for the language LANG\_GERMAN:

1. ./gradlew ducxAddTranslation --target LANG\_GERMAN --source LANG\_ENGLISH

This task can be executed on solution level or on project level.

ducxBuild

Compile the project or the whole solution.

To reduce execution time , you can restrict the compilation to Java code only by using the ‑‑java‑only option.

This task can be executed on solution level or on project level.

ducxBuildLocal

Compile the project or the whole solution with assigning object addresses to new component objects.

To reduce execution time it is possible to restrict the compilation to the java code using the option ‑‑java‑only.

This task can be executed on solution level or on project level.

ducxLoad

Upload the solution. If the system is a development system, a delta load including validation is performed to update the solution.

Following options are available for task ducxLoad:

--java-only uploads java only.

--no-java-only disables option --java-only.

--validate-solution-repository validates the solution repository on delta upload.

--no-validate-solution-repository disables the validation at upload.

--rerun causes the task to be re-run even if up-to-date.

ducxGenerate

Generates the required libraries or source code for language integrations without generating the software component.

ducxGenerateOpenAPI

Generates OpenAPI clients and servers based on a specification.

The option --specification is required and specifies the path to the OpenAPI specification.

The option --server specifies that a server should be generated. Additionally, it requires a valid app.ducx reference to be passed.

The option --serverimpl specifies that a server implementation should be generated.

The option --client specifies that a client should be generated. Additionally, it requires a valid app.ducx reference to be passed.

**Note:** It is required to either pass--serveror--client.

The names of the created files are derived from the reference specified for --server or --client.

Example: ../../gradlew ducxGenerateOpenAPI   
 --specification=resources/petstore.yaml  
 --server=PetstoreServer  
 --serverimpl

ducxPatchProject

Adds java settings to a project when called with ‑‑add‑java. To overwrite existing entries or files use the option ‑‑force.

ducxStartCoverage/ducxStopCoverage

Start and stop coverage sessions with the gradle tasks ducxStartCoverage and ducxStopCoverage, operating on single projects or the whole solution. After ducxStopCoverage, each software component folder contains coverage information.

Display the coverage with the VS Code extension “Coverage Gutters”.

**Note:** To access the latest Gradle tasks, update the Gradle version in your project's settings.gradle file. The current Gradle version is 1.187.0.

ducxRunTests

Executes the specified tests on the server and creates a jUnit XML report and an html report.

The –ref and, --tags options specify which tests should be executed.

You can specify both --refs and --tags in the same request and all unit tests that match either will be executed. Note that some tests may be executed multiple times, if there are overlaps. If you specify neither of them, then it will automatically specify the SWC of the current project (if any) and all subprojects. If you instead want to execute all tests on the server, you can specify --executeAll.

The --refs option accepts a comma-separated list of references to SWCs, unit tests, or unit test groups. References may be fully qualified or unqualified—the latter runs all unit tests or groups matching that reference. If an SWC is specified, all of its unit test groups will be executed.

The option --tags accepts a comma-separated list of tags. All unit tests/unit test groups with one of these tags will be executed.

The option --jUnitFile is a (optionally relative) path to a result jUnit XML report file that should be created. The default is " test-results/app.ducx/unittests.xml".

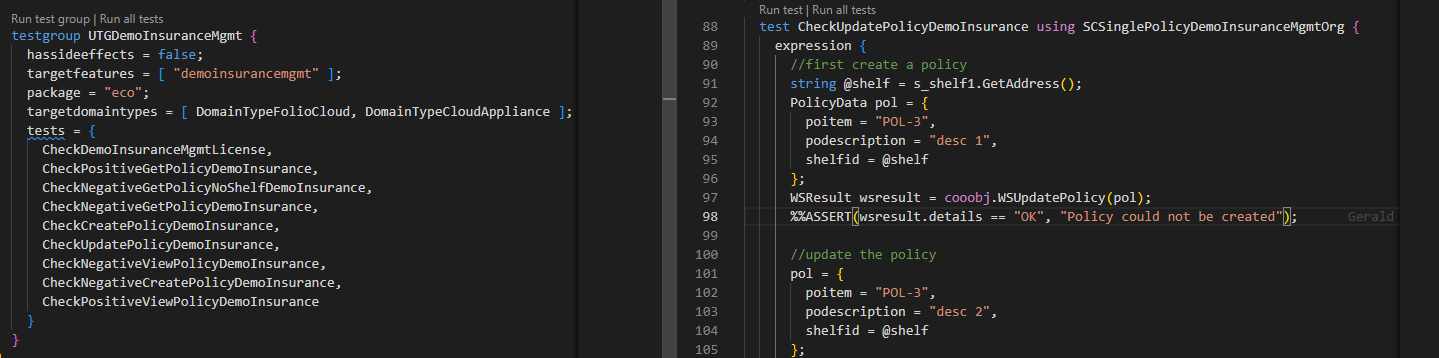
The option --htmlFile is a (optionally relative) path to a result HTML report file that should be created. The default is " test-results/app.ducx/unittests.html".

The option --executeAll can be used to execute all tests on the server. If this option is specified all given --refs and --tags are ignored.

The option --xslTransformation is optional and specifies an XSL transformation to create the HTML report.

## Execute Unit Tests

You can execute unit tests directly from the editor. Options include executing a single test, all tests, a specific test group, or an individual scenario. Before running any unit tests, ensure that your .ducxws file is properly configured.



Debugging unit tests is straightforward: start the debugger and then execute the test you want to debug.

## Code Coverage

You can evaluate code coverage for Fabasoft app.ducx files using the [Coverage Gutters](https://marketplace.visualstudio.com/items?itemName=ryanluker.vscode-coverage-gutters) extension in Visual Studio Code. Once the extension is installed, follow these steps:

1. Start the Coverage Process  
   Run the Gradle task: ducxStartCoverage.
2. Execute Your Source Code  
   Perform the actions or run the code you wish to analyze for coverage.
3. Stop the Coverage Process  
   Run the Gradle task: ducxStopCoverage.

Enable the "watch coverage" feature from the Coverage Gutters extension to easily visualize covered and uncovered lines directly in your code.

**Note:** Code coverage is currently unavailable while debugging.

## Evaluate Expression

To execute expressions in Fabasoft app.ducx files:

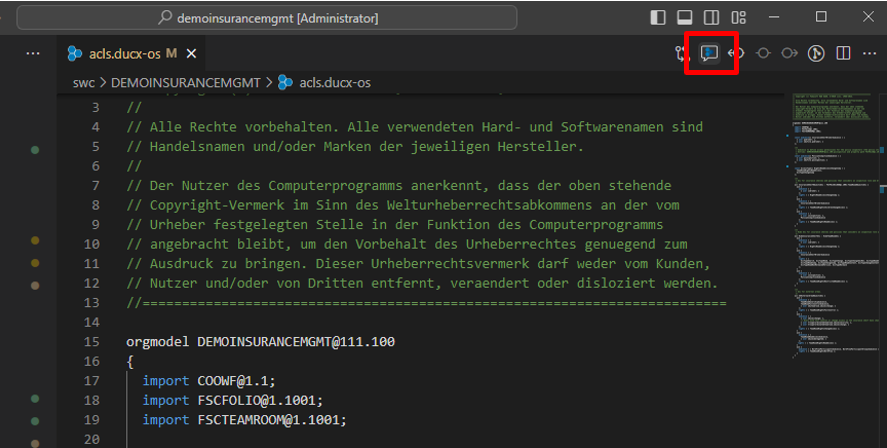
1. Select the expression block you want to evaluate.
2. Use the “Evaluate Expression” command.

You can trigger this command in one of the two ways, either press the shortcut Alt + X or open the Command Palette in VS Code, then search for and select app.ducx: Evaluate Expression.

The output of the evaluated expression will appear in the OUTPUT panel in VS Code.

## Feedback

The Fabasoft app.ducx extension includes a feedback button. To use it, open the app.ducx file you are having issues with and click the feedback button icon located in the top-right corner of Visual Studio Code.



We encourage you to use the feedback button to report any issues. Your feedback will help us improve and enhance your development experience with the extension.

## Support for Large Workspaces

The Maximum Number Of Projects To Process On Launch setting in the Fabasoft app.ducx extension limits how many projects can be processed automatically upon launch (default: 15). If the number of projects in the workspace exceeds this limit, none of the projects will be processed.

Processed projects are handled by the app.ducx extension, while other projects remain inactive. When a file from an inactive project is opened, the entire project is automatically processed as well. Projects stay processed as long as VSCode is running.

For inactive projects an info diagnostic is generated. Additionally, upon launch, if there are any projects in the workspace that were not processed, a pop-up message will appear, offering an option to process all projects manually or disabling the Show Dialog Maximum Number Of Projects To Process On Launch setting by clicking on “Do not show again” button.

