

Enterprise Analyzer チュートリアル

生成 AI 連携 編

1 目的

生成 AI と静的コード分析の力を組み合わせることで、開発者は複雑なソースコードでもすぐに理解できるようになり、時間を短縮してメンテナンス時間を加速できます。

本チュートリアルでは Enterprise Analyzer に生成 AI を組み合わせることで COBOL のソースコードの解析を容易にできるその機能と設定手順について説明しています。

2 前提

- 2.1 本チュートリアルで使用した OS: Windows Server 2022 Standard Edition
- 2.2 使用マシンに Enterprise Analyzer 11.0 がインストールされていること
- 2.3 Enterprise Analyzer 11.0 基本機能の紹介と利用方法チュートリアルが完了していること
- 2.4 Enterprise Analyzer 11.0 クエリー作成編チュートリアルが完了していること
- 2.5 生成 AI の製品として Open AI もしくは Ollama のインストールと設定が完了していること

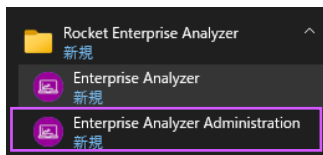
3 チュートリアル手順の概要

- 3.1 Enterprise Analyzer Administration の起動
- 3.2 Open AI を利用した生成 AI の設定
- 3.3 Ollama を利用した生成 AI の設定
- 3.4 Enterprise Analyzer の起動と AI 機能の表示
- 3.5 Chat ウィンドウから AI 機能の動作確認
- 3.6 ファイルやデータ項目に関する基本情報を表示
- 3.7 ファイルやプログラムの機能的な関係を調査
- 3.8 コード品質を分析
- 3.9 プログラムの制御フローを表示
- 3.10 VS Code のインストールと設定
- 3.11 Rocket Analyzer の設定
- 3.12 VS Code 上でプログラムサマリーを生成
- 3.13 VS Code 上でプログラムフローを生成

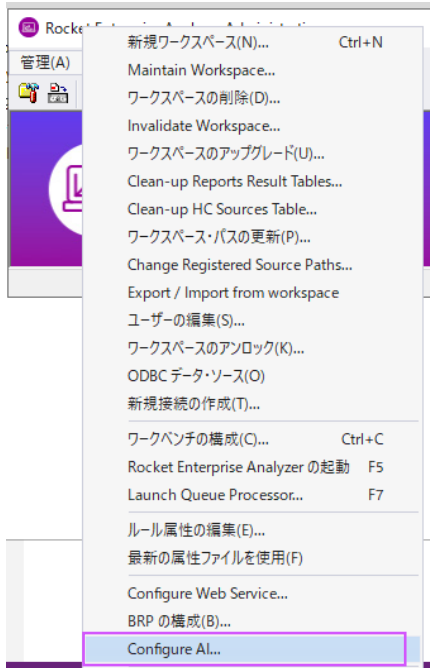
3.1 Enterprise Analyzer Administration の起動

生成 AI 利用のための設定を行います。

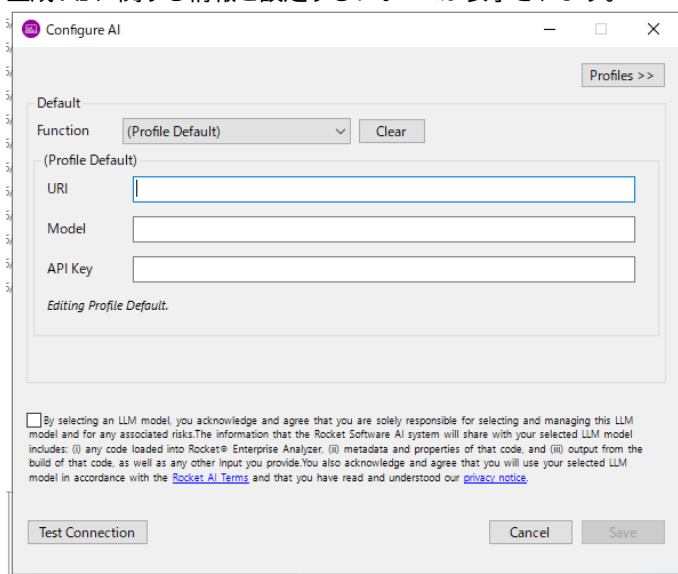
3.1.1 Windows メニューから [Enterprise Analyzer Administration] を起動します。



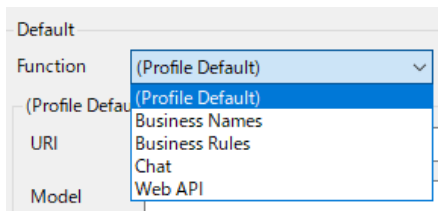
3.1.2 [管理]メニュー > [Configure AI...] を選択し、AI の設定画面を表示します。



3.1.3 生成 AI に関する情報を設定するフォームが表示されます。



デフォルトの生成 AI の設定以外に機能別に複数の生成 AI を指定することが可能です。今回は「Profile Default」のまま全ての機能に単一の生成 AI を指定します。



3.2 Open AI を利用した生成 AI の設定

OpenAI 社は、複数のグローバルリーダーによって設立された非営利研究機関です。人類にとって有益な AI 発展を目的とし、さまざまな AI モデルやサービスを開発しています。

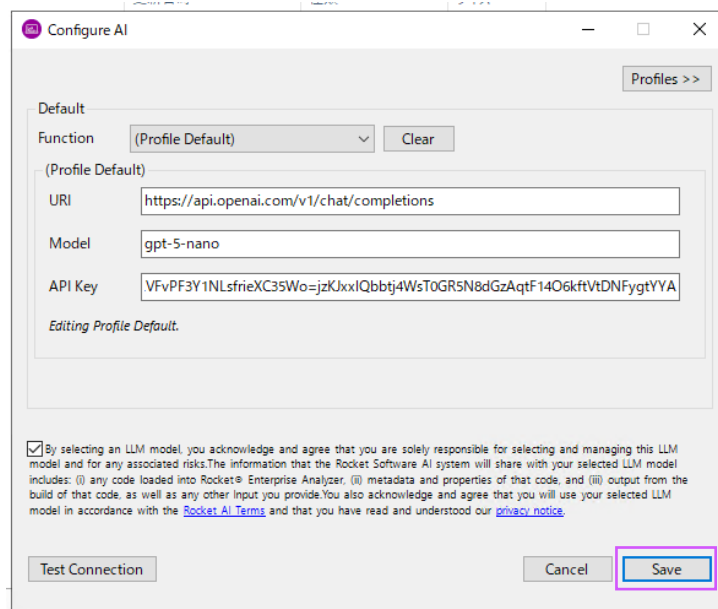
3.2.1 それぞれのフィールドは、下記のように入力し、[Save]ボタンを押して保存してください。

[Function]: “Profile Default”

[URI]: <https://api.openai.com/v1/chat/completions> を入力

[Model]: “gpt-5-nano”もしくは“gpt-4.1”ma を入力

[API Key]: 自社の Open AI 契約管理者から適正なキーを入手して入力
生成 AI に関する許諾にチェック



3.3 Ollama を利用した生成 AI の設定

Ollama は、ローカル環境で生成 AI のエンジンを動かすためのプラットフォームです。この製品を使うとローカル環境で生成 AI とやり取りすることが可能になります。※ローカル環境に生成 AI を用意して解析を行う場合、かなりの高性能 PC が必要となりますのでパフォーマンスの観点から推奨いたしません。

3.3.1 最初に Ollama が利用できるように設定を行ってください。詳細は Ollama のサイトもしくはインターネット上で多数見受けられます。

3.3.2 <https://ollama.com> より、プラットフォームに対応したインストーラーをダウンロードします。

3.3.3 Ollama のコマンドラインより、ollama pull mistral とタイプして「mistral」モデルをダウンロードします。「gemma3:12b」モデルも使用可能です。

3.3.4 それぞれの欄には下記のように入力し、[Save]ボタンを押して保存してください。

[URI]: <http://localhost:11434/v1/chat/completions> を入力

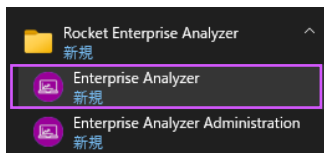
[Model]: “mistral”を入力

[API Key]: 未入力

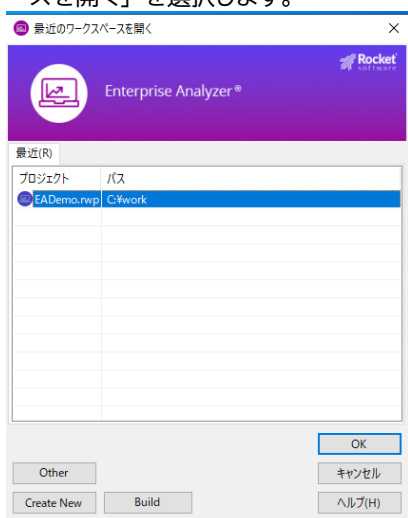
3.4 Enterprise Analyzer の起動と AI 機能の表示

Enterprise Analyzer のワークベンチを起動して Chat 画面にていくつかの問い合わせを行ってみます。

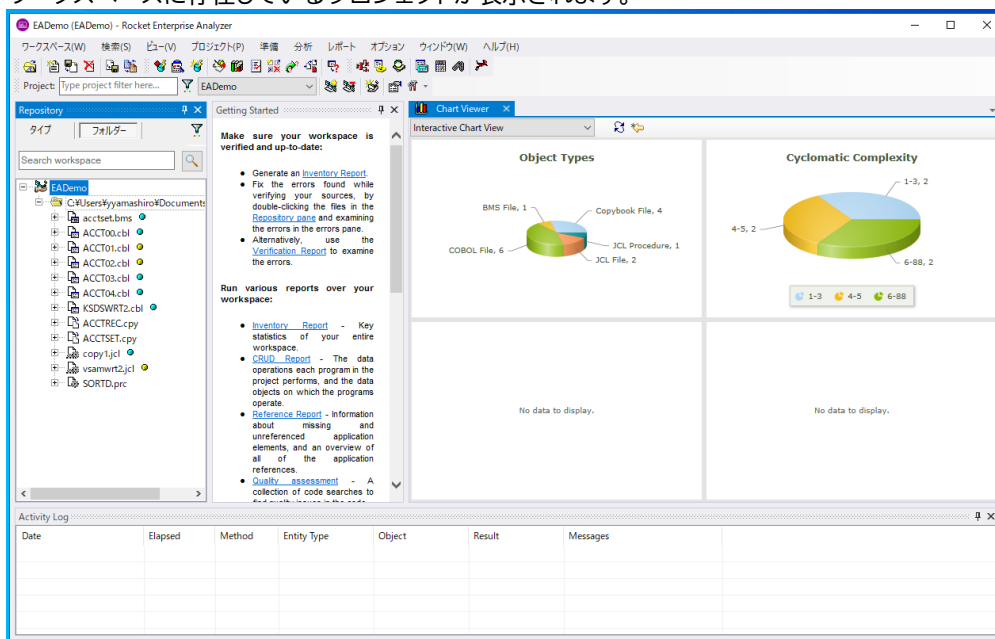
3.4.1 Windows メニューから [Enterprise Analyzer] を起動します。



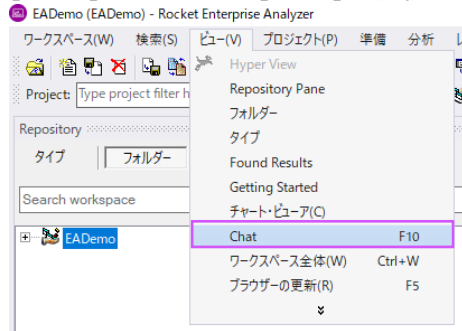
3.4.2 起動と同時に「ワークスペースを開く」ウィンドウが表示されますので、使用するワークスペースを選択して [OK] ボタンをクリックします。もし、閉じている場合は、メニューから [ワークスペース] メニューの [ワークスペースを開く] を選択します。



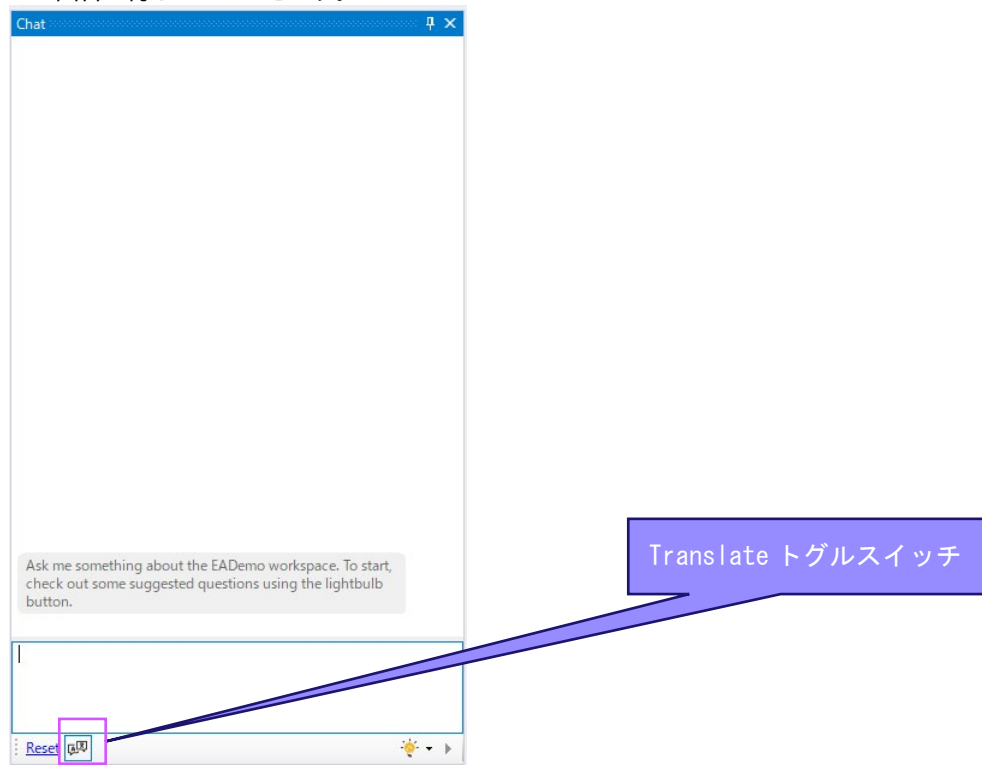
3.4.3 ワークスペースに存在しているプロジェクトが表示されます。



3.4.4 [ビュー] メニューから [Chat] を選択します。



3.4.5 [Chat] ペインがワークベンチ内に表示されます。[Translate] トグルスイッチを押すことで日本語による問い合わせと回答を得ることができます。



3.5 Chat ウィンドウから AI 機能の動作確認

以下の前提条件が満たされていることを確認してください。

- ・製品のインストール時に、Enterprise Analyzer の AI 機能を使用することに同意するチェックボックスにチェックを入れてインストールを行なっていること。このチェックボックスにチェックを入れていない場合、Enterprise Analyzer の UI に [Chat] ウィンドウは表示されない。
- ・Enterprise Analyzer ワークスペースを設定し、ソースコードを登録および検証済みである。

3.5.1 Chat ウィンドウ内に自然言語で入力することにより、リポジトリ内にある情報を AI が検索して結果を表示します。 ※[Translate] トグルスイッチを押すことで日本語による問い合わせが可能です。

問い合わせの目的	問い合わせ例
ファイルやデータ項目に関する基本情報を表示する	・What is ProgramA.cbl? ・What is my-data-item?
ファイルやプログラム間の機能的な関係を調査する	・What does MyJob.jcl reference?

	•What does MyProgram.cbl call? •What does MyProgram.cbl include? •What does MyProgram.cbl write?
ファイルやデータ項目が参照されている場所を特定する	•What references MyProgram.cbl? •What calls MyProgram.cbl? •Where is my-var referenced in MyProgram.cbl?
コード品質を分析し、コード検索を実行する	•Are there any code quality problems with MyProgram.cbl? •Show dead code in MyProgram.cbl •Show unused data in MyProgram.cbl
プログラムの制御フローを表示する	•Show program flow for MyProgram.cbl.

3.6 ファイルやデータ項目に関する基本情報を表示する

- 3.6.1 [Chat]ウィンドウ内に下記のように入力を行います。
「ACCT01.cbl は何ですか？」

[Chat]ウィンドウ内に「ACCT01.cbl」の概要が表示されます。

[ACCT01.cbl](#) は COBOL ファイルです。

- ソース行数は 336 行です。
- 使用されている方言は Rocket です。
- プログラムは CICS 対応です。

3.7 ファイルやプログラムの機能的な関係を調査する

- 3.7.1 [Chat]ウィンドウ内に下記のように入力を行います。
「ACCT01.cbl が参照しているファイルを教えてください。」

[Chat]ウィンドウ内に「ACCT01.cbl」が参照しているファイルが表示されます。

[ACCT01.cbl](#) は 13 のエンティティを参照します: [AC01](#) [AC02](#) [AC03](#) [ACCT](#) [ACCT01.ACCTFIL](#) [ACCT01.ACCTIX](#) [ACCTMENU](#) [ACCTREC.cpy](#) [ACCTSET.ACCTDTL](#) [ACCTSET.ACCTMNU](#) [ACCTSET.cpy](#) [DFHAID.CPY](#) [DFHBMSCA.CPY](#)

[Show References results](#)

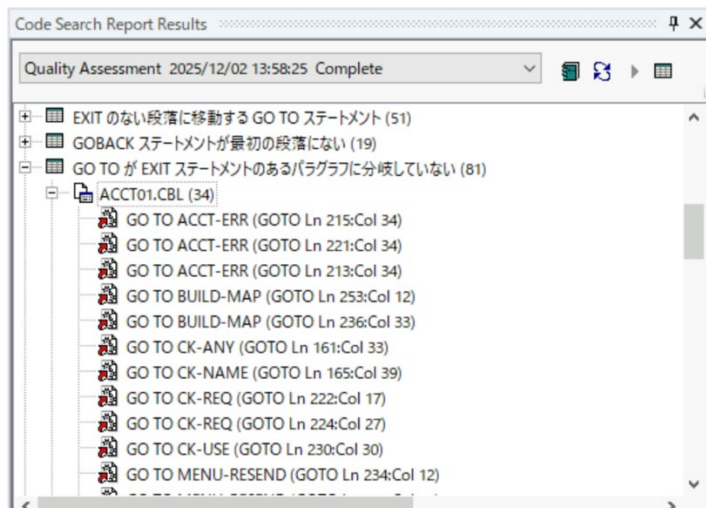
3.8 コード品質を分析

- 3.8.1 [Chat]ウィンドウ内に下記のように入力を行います。
「ACCT01.cbl はコード品質に問題ありますか？」

コード品質に関する情報が表示されます。[Show Code Search results]をクリックすることで[Code Search Report Results]パンに詳細な情報が表示されます。

'ACCT01.cbl' は1135件の品質関連の問題があり、18の異なるカテゴリに分類されています。

[Show Code Search results](#)

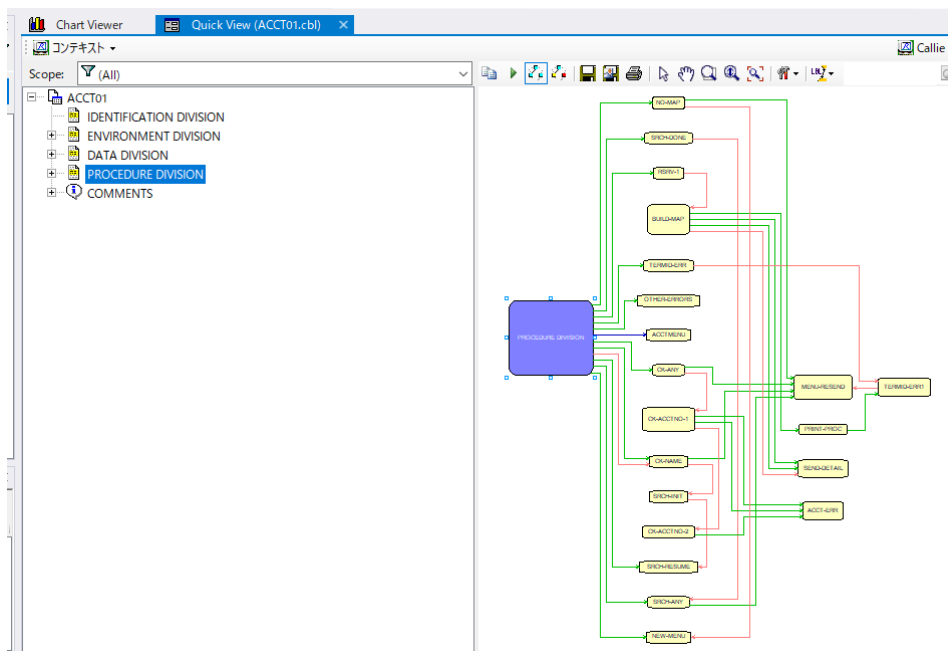


3.9 プログラムの制御フローを表示

- 3.9.1 [Chat]ウィンドウ内に下記のように入力を行います。
「ACCT01.cbl のプログラムフローを表示してください。」

プログラムフローの作成が行われ、[Quick View] ペインにプログラムフローが表示されます。

ACCT01.cbl のプログラムフローを開始します。

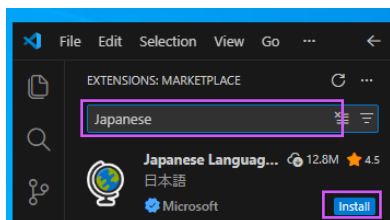
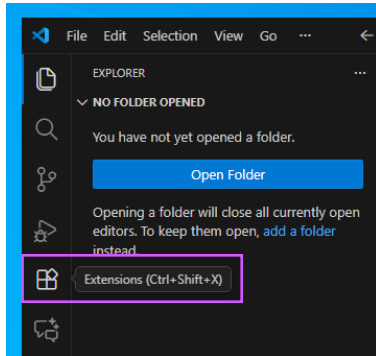


3.10 Visual Studio Code のインストールと設定

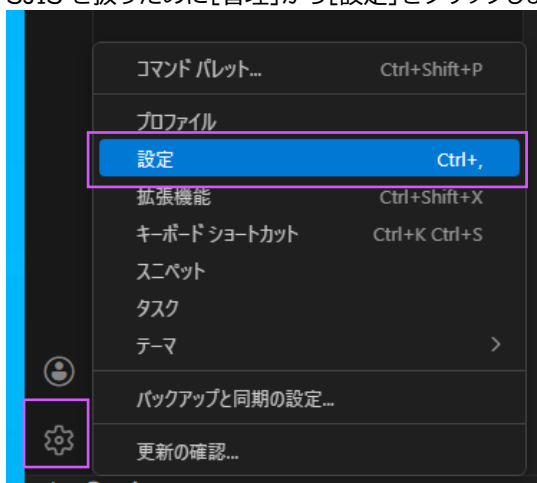
Visual Studio Code に Rocket COBOL のプラグインを入れることで COBOL コードの編集ができるようになりますが、それに加えて Enterprise Analyzer のプラグインを入れることで編集中の COBOL コードの解析も行うことができるようになります。ここでは、生成 AI を利用した COBOL のコード解析の手順を説明します。

3.10.1 マイクロソフトのサイトより Visual Studio Code をインストールします。

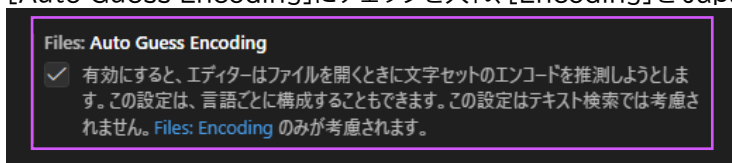
3.10.2 日本語ランゲージパックをインストールして、GUI を日本語化します。[Extensions]をクリックし、"Japanese"とタイピングしてフィルタリングし、Japanese Language Pack の[Install]をクリックします。

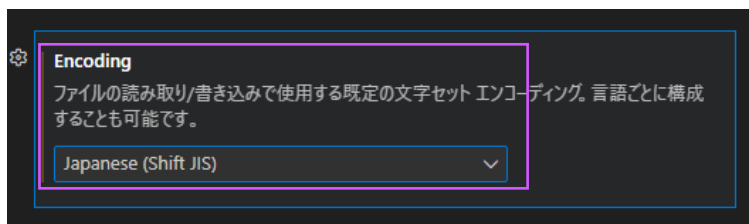


3.10.3 SJIS を扱うために[管理]から[設定]をクリックします。

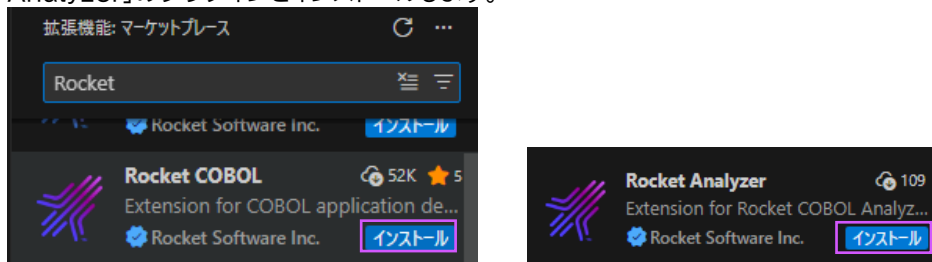


3.10.4 [Auto Guess Encoding]にチェックを入れ、[Encoding]を Japanese (SJIS)に変更します。



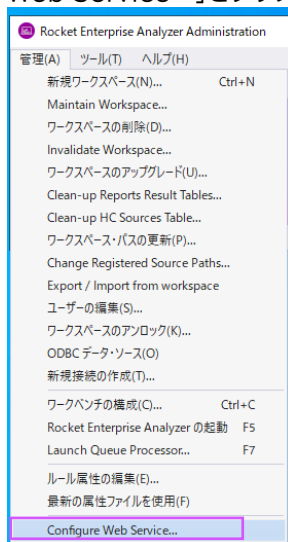


3.10.5 [エクステンション]をクリックし、「Rocket」とタイピングしてフィルタリングし、「Rocket COBOL」と「Rocket Analyzer」のプラグインをインストールします。

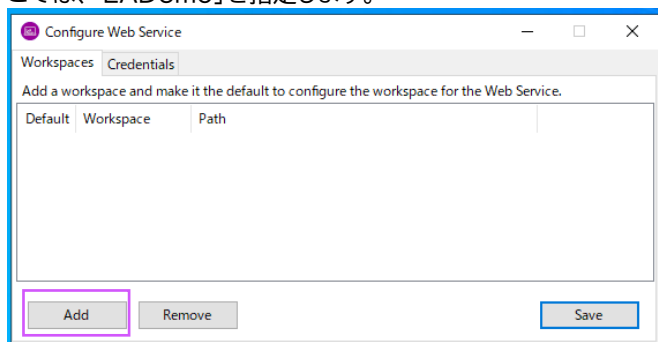


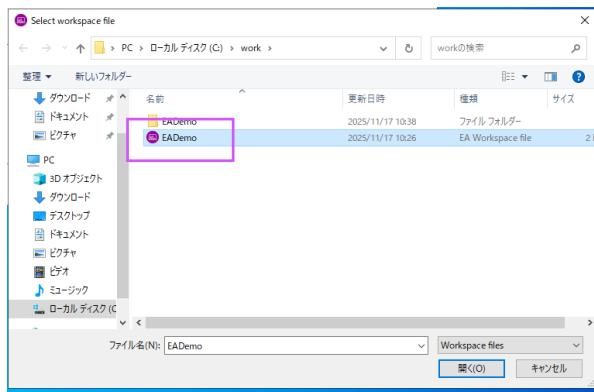
3.11 Rocket Analyzer の設定

3.11.1 Windows メニューから[Enterprise Analyzer Administration]を起動し、[管理]メニュー > [Configure Web Service...]をクリックします。

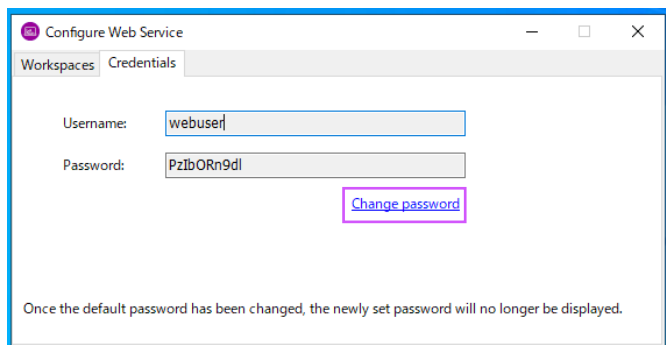
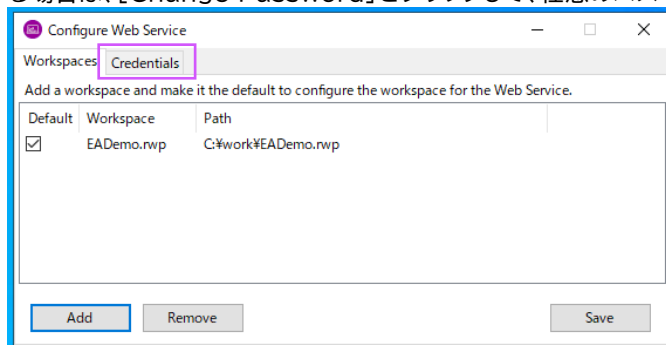


3.11.2 [Configure Web Service]画面が表示されるので、[Add]ボタンをクリックし、Workspace を指定します。ここでは、「EADemo」を指定します。

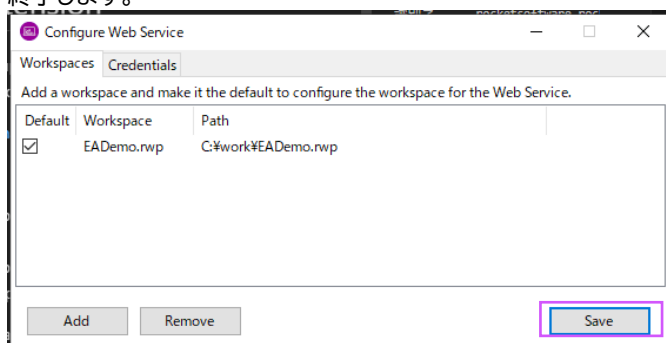




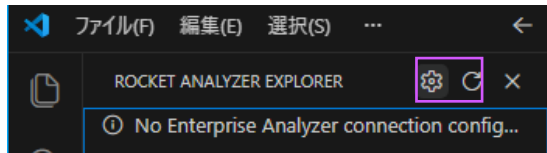
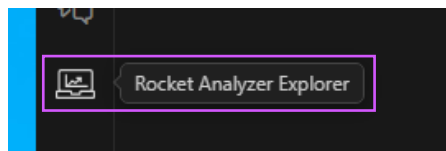
3.11.3 次に[Credentials]タブをクリックし、リポジトリへのログインアカウントを記録しておきます。パスワードを変更する場合は、[Change Password]をクリックして、任意のパスワードを指定してください。



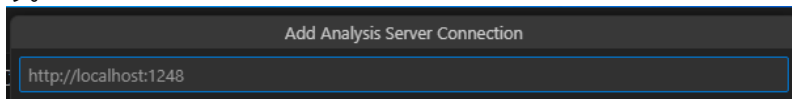
3.11.4 [Workspace]タブをクリックし、[SAVE]ボタンを押して、[Enterprise Analyzer Administration]の画面を終了します。



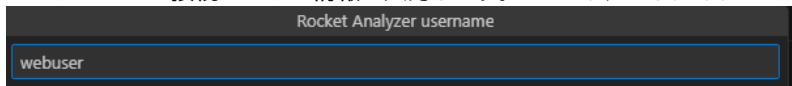
3.11.5 VS Code の画面に戻り、[Rocket Analyzer Explorer]をクリックし、[Configure Connection]のアイコンをクリックします。



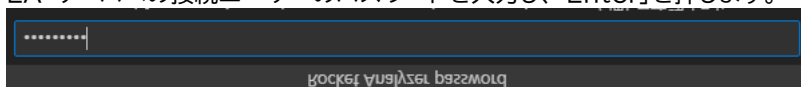
3.11.6 EA サーバへの接続アドレスを入力します。ここでは、"http://localhost:1248"とタイプして、「Enter」を押します。



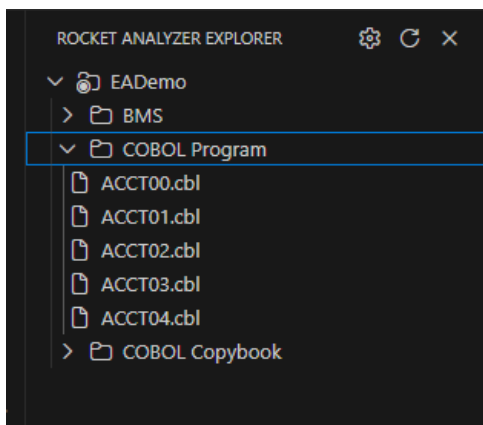
3.11.7 EA サーバへの接続ユーザー情報を入力します。ここでは、"webuser"とタイプして、「Enter」を押します。



3.11.8 EA サーバへの接続ユーザーのパスワードを入力し、「Enter」を押します。

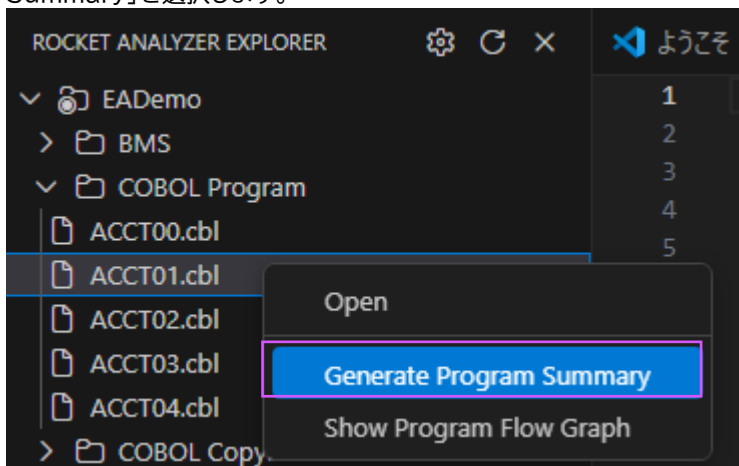


3.11.9 [ROCKET ANALYZER EXPLORER]に「EADemo」の情報が展開されます。



3.12 VS Code 上でプログラムサマリーを生成

3.12.1 「Rocket Analyzer Explorer」から「ACCT01.cbl」を選択し、マウスの右クリックから[Generate Program Summary]を選択します。



3.12.2 AI で解析を行い、このプログラムに関するサマリー情報、属性、リレーションダイアグラム、プログラムコントロールフローダイアグラム等が表示されます。

ACCT01.cbl

Enterprise Analyzer Summary: ACCT01.cbl

Program Summary: ACCT01.cbl

ACCT01 is a COBOL program defined in the source file ACCT01.cbl that runs under CICS. It is the first program invoked by the AC01 transaction and serves as a controller that analyzes all incoming requests. It handles name inquiries and record displays directly; for update transactions it displays the appropriate data-entry screen and sets the next transaction identifier to AC02 to complete the update, while for print requests it starts the AC03 transaction to perform the actual printing.

The program performs 11 file operations and interacts with two main data files: ACCTFIL and ACCTIX. It reads account records from ACCTFIL using a RIDFLD(ACCTC) path and uses ACCTIX for search results via STARTBR DATASET(ACCTIX), READNEXT, and ENDBR sequences. It also maintains structured working storage for search controls and data transformation, and relies on copybooks ACCTREC and ACCTSET along with the DFHAID and DFHBMSCA copies for data definitions.

ACCT01 exchanges data with users through BMS screens via maps ACCTSET.ACCTMNU and ACCTSET.ACCTDTL. It employs CICS SEND MAP and RECEIVE MAP operations to render and collect screen data, and uses queues such as USE-QID/USE-REC to coordinate data entries for updates and validations. For printing, it initiates the AC03 transaction and, in update scenarios, routes control to the next transaction AC02 after data entry. The program also uses an XCTL call to transfer control to the program defined in ACCTMENU when appropriate.

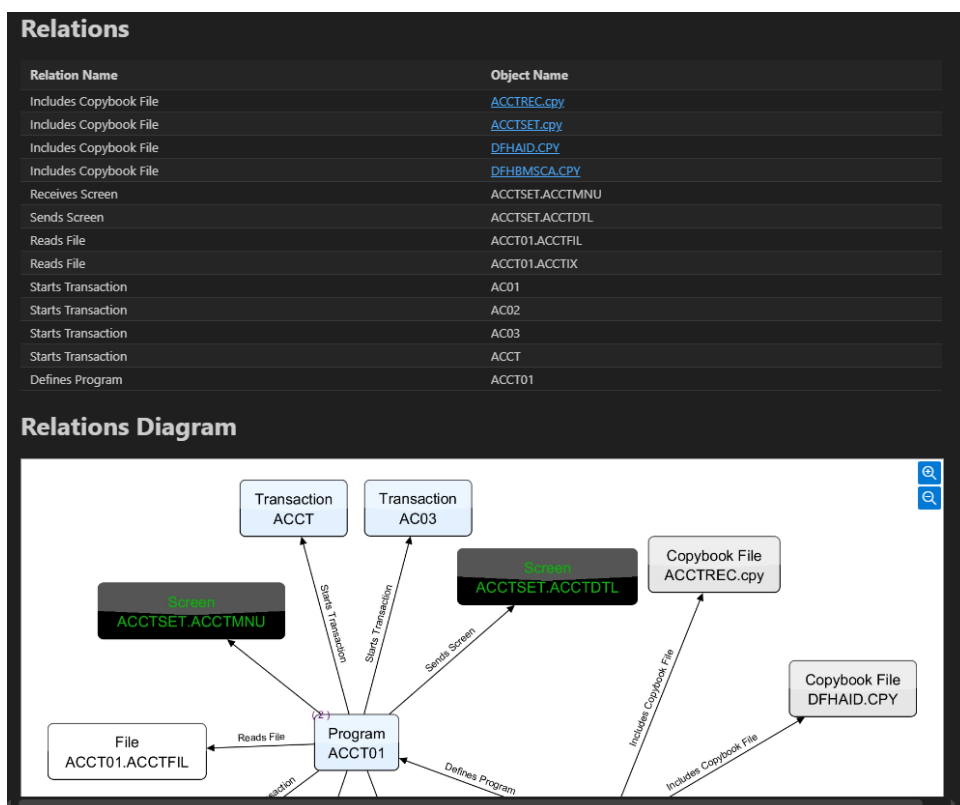
From a quality perspective, ACCT01 has a cyclomatic complexity rating of 45, with 27 go-to statements, indicating a relatively complex and potentially hard-to-maintain flow. It performs 11 file operations and contains 336 source lines, which places it below the workspace average in maintainability. The source includes eight commented lines, which provide some inline context but do not fully mitigate the overall complexity of the codebase.

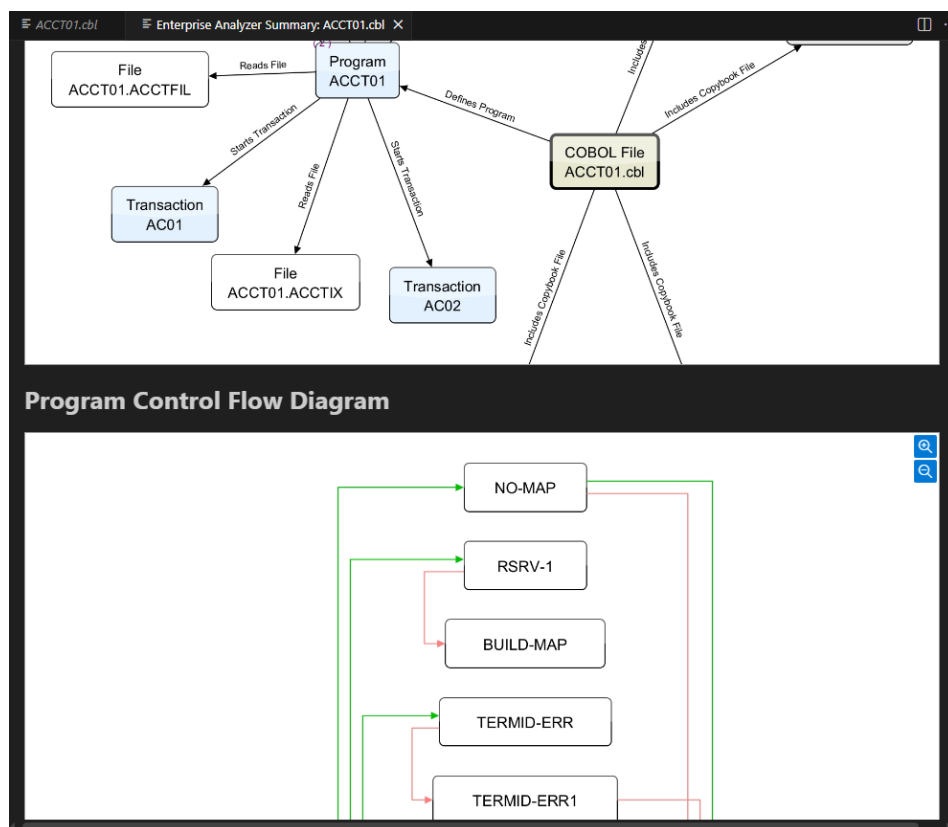
Technical and structural details include the use of the Rocket COBOL dialect, and the inclusion of copybooks ACCTREC.cpy, ACCTSET.cpy, DFHAID.CPY, and DFHBMSCA.CPY. The program defines a message list (MSG-LIST) with a REDEFINES structure (MSG-TEXT) to support user messages, and uses the DFHCOMMAREA in the LINKAGE SECTION for inter-program communication. It relies on a range of CICS commands (HANDLE CONDITION, START, READ, WRITEQ TS, SEND/RECEIVE MAP, ENDBR, RETURN TRANSID) to coordinate data flow, user interaction, and screen rendering, with explicit paths for error handling and normal operation.

This content was created by AI and may include errors. See [link](#) for details.

Attributes

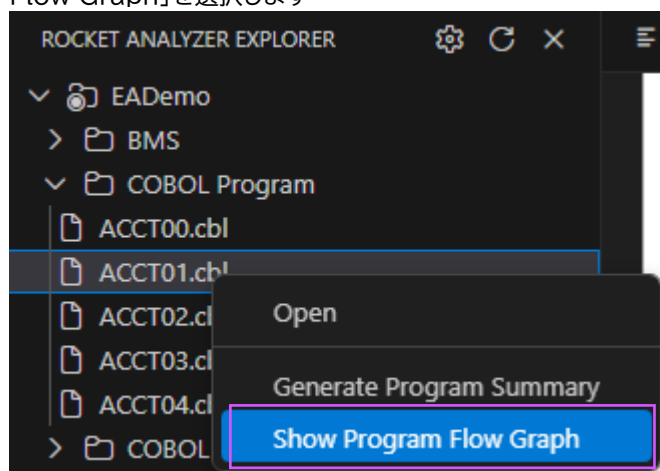
Name	Value
Lines Of Code	1,174
Cyclomatic Complexity	45
Dead Lines	220



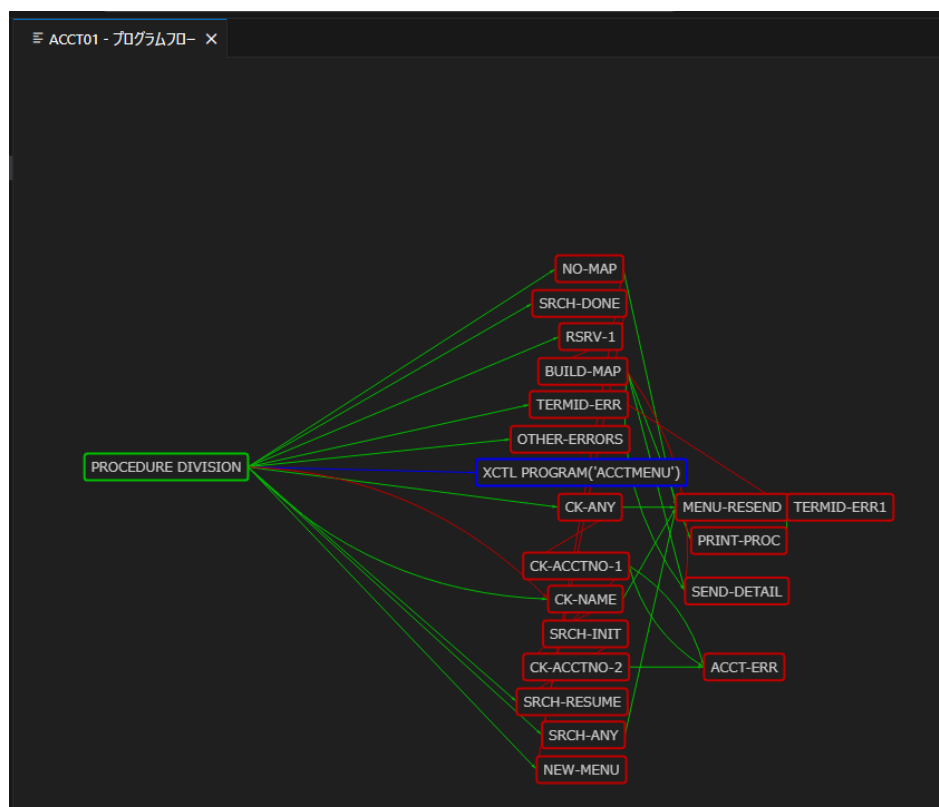


3.13 VS Code 上でプログラムフローを生成

3.13.1 「Rocket Analyzer Explorer」から「ACCT01.cbl」を選択肢、マウスの右クリックから[Show Program Flow Graph]を選択します



3.13.2 AI で解析を行い、プログラムフロー図が表示されます。



4 免責事項

本チュートリアル の例題ソースコードは機能説明を目的としたサンプルであり、無謬性を保証するものではありません。例題ソースコードは弊社に断りなくご利用いただけますが、本チュートリアルに関わるすべてを対象として、二次的著作物に引用する場合は著作権法の精神に基づき適切な扱いを行ってください。

本チュートリアルで学習した技術の詳細については製品マニュアルをご参照ください。