Rocket® software

Enterprise Developer を使用した COBOL 開発業務のモダナイゼーション

Version.2
Last updated: 2026 年3月

開発業務のモダナイゼーション

Rocket software

目次

各機能の詳細に関しては、製品マニュアルページからご利用になるバージョンを選択後、内容をご確認ください。
https://www.amc.rocketsoftware.co.jp/support/manuals.asp

1. はじめに

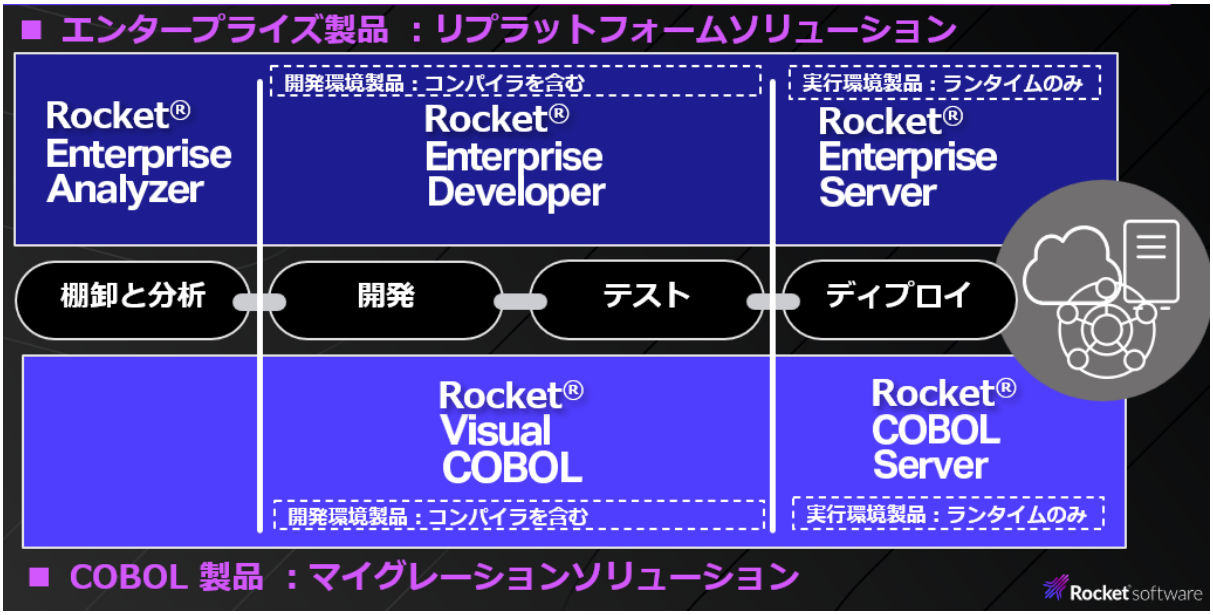
IBM メインフレームで稼働する基幹システムは、ビジネスに欠くことのできない重要な経営資産として現在も世界のビジネストランザクションの主要な業務を担い、日々稼働し続けています。
しかし、開発業務には IBM メインフレームの多くの資源が必要となり、これによるコストの増加や、古い開発スタイルにより、作業の効率化が図りづらいなどの問題があることも事実です。
例えば、クラウドなどのオープン環境へ基幹システムを移行した場合、開発スタイルはどのように変化するのでしょうか。

本書では、弊社のモダナイゼーション支援製品である Enterprise Developer を使用した、リプラットフォーム後の COBOL アプリケーションの開発スタイルについて解説します。

2. モダナイゼーション支援製品の分類

弊社のモダナイゼーション支援製品は、過去の投資を無駄にせず、実績のある COBOL、PL/I アプリケーションをオープン環境で再活用できる開発/実行環境製品です。

製品は2つに分類されており、エンタープライズ製品群には、静的解析ツールの Enterprise Analyzer、コンパイラと開発用実行環境を含む開発環境製品の Enterprise Developer、ランタイムのみを含む実行環境製品の Enterprise Server が含まれ、COBOL と PL/I アプリケーション、IBM メインフレームのミドルウェア互換機能をサポートするリプラットフォームを支援します。
COBOL 製品群は、コンパイラと開発用実行環境を含む開発環境製品の Visual COBOL と、ランタイムのみを含む実行環境製品の COBOL Server があり、IBM、国産メインフレーム、オープンレガシーからの COBOL アプリケーションの移行を支援します。

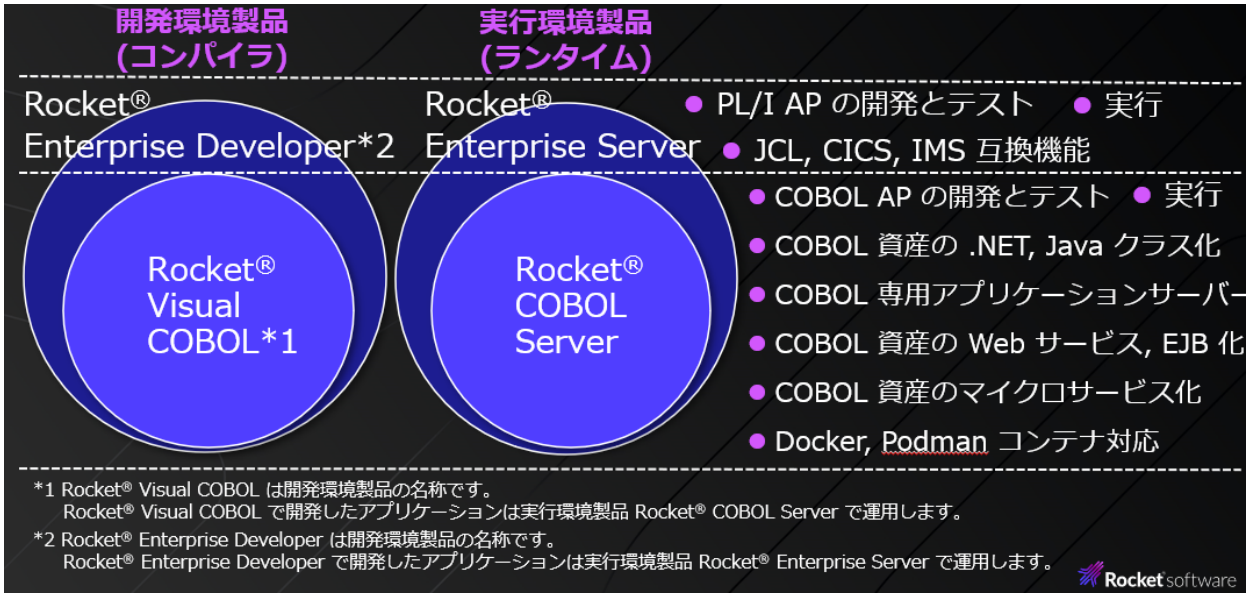


3. モダナイゼーション支援製品の関係性

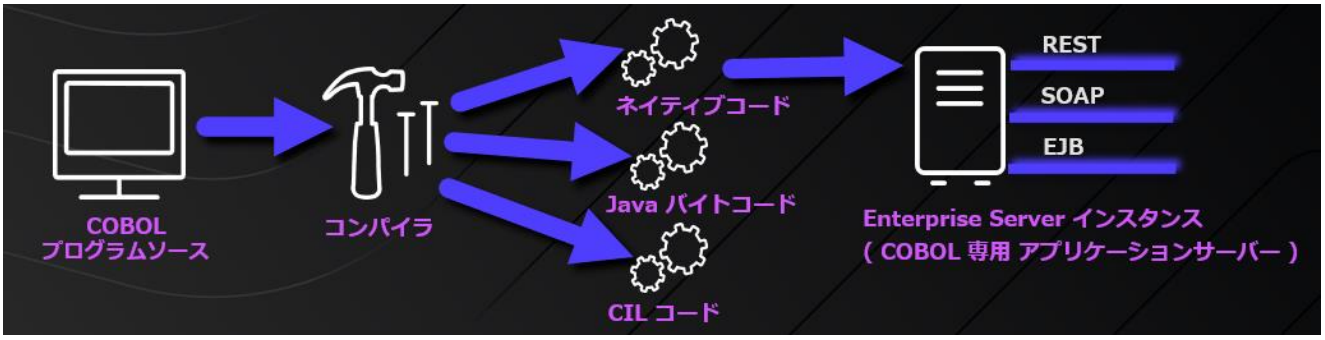
エンタープライズ製品は COBOL 製品の全機能を含む上位製品となり、リプラットフォーム後も新しい技術を取り入れながら、更なるモダナイゼーションを目指すことができます。

4. モダナイゼーション支援製品の機能分布

既存の資産を将来に渡り有効利用できるように、COBOL や PL/I アプリケーションを新しい技術と連携させる機能が製品には含まれています。

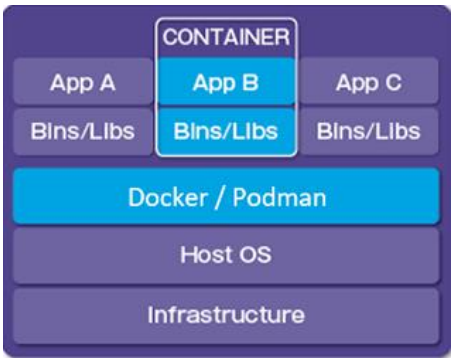


COBOL製品では、弊社のコンパイラ技術により、COBOLソースからネイティブコード、Javaバイトコード、CILコードの3種類を生成することができます。

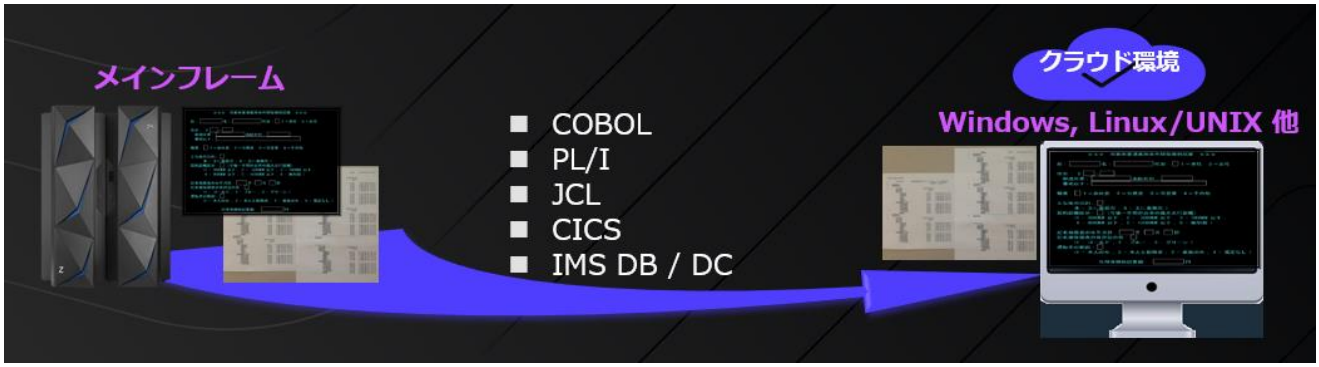


生成されたネイティブコードは、製品に含まれるCOBOL専用アプリケーションサーバーを介したREST、SOAP形式のWebサービスの展開とEJB連携を実現することができます。これにより、様々なデバイスからCOBOLのロジックを呼び出すことが可能になります。
Javaバイトコードは JVM 上で、CIL コードは.NET 上で他のアプリケーションとシームレスに連携しながら稼働させることができます。

また、各製品はクラウド上でマイクロサービスを導入する際に欠かせないコンテナ型仮想化にも対応しています。

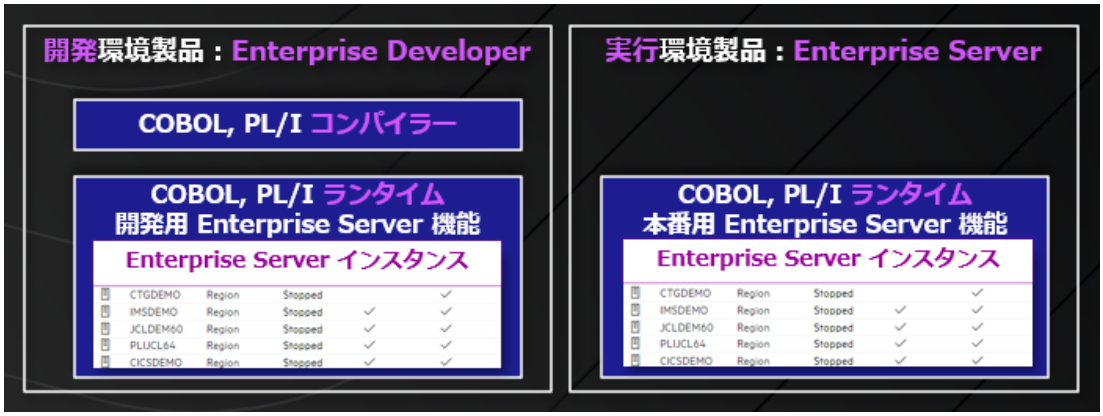


エンタープライズ製品は、これらの機能に加えて、PL/I 言語のサポートや IBM メインフレームの JCL、CICS、IMS 互換性機能を持ち、オープン環境へ移行後も JCL、EXEC CICS、EXEC DLI、CBLTDLI、PLITDLI 構文などを利用することができ、必要最低限のコストと期間でアプリケーションの品質を損なうことなく、IBM メインフレームからのリプラットフォームを実現することができます。



5. 製品の利用用途

COBOL、PL/I ランタイムのみを含む実行環境製品の Enterprise Server は、コンパイルを必要としない本番環境に使用する製品です。アプリケーションを実行させる単位である Enterprise Server インスタンスは、例えば JCL を対象としたバッチ用、IMS、CICS などのオンライン用など、運用用途に合わせて複数設定することができます。



開発環境製品の Enterprise Developer は Enterprise Server と同等の機能を持つ開発用実行環境を含んでおり、IBM メインフレームのミドルウェア互換機能を利用した単体テストを実行することができます。

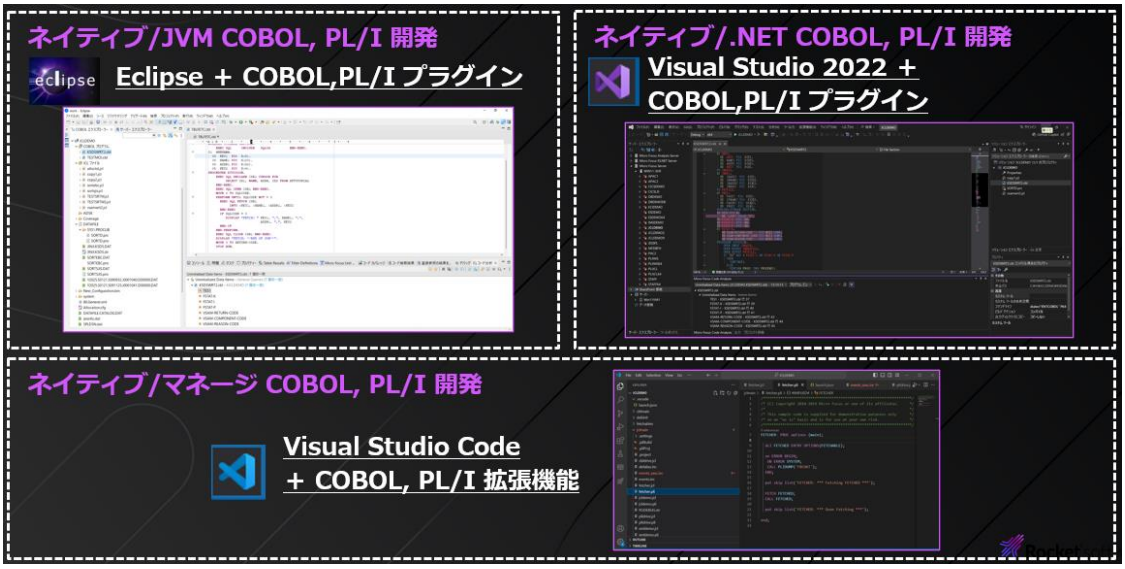
6. Enterprise Developer を利用した開発業務

COBOL アプリケーションをオープン環境へ移行後は、開発環境製品である Enterprise Developer を使用して開発業務を行います。Enterprise Developer が持つ様々な機能を利用することにより、作業の効率化が計れ、開発用実行環境を利用することにより、IBM メインフレームを独り占めするイメージで単体テストを実施することができます。

この章では、Enterprise Developer の具体的な開発機能について説明します。PL/I の開発機能については、「PL/I 開発業務のモダナイゼーション」ホワイトペーパーをご参照ください。また、リプラットフォームにおける一般的な手順や注意点については「COBOL、PL/I アプリケーションのリプラットフォーム手順と注意点」ホワイトペーパーをご参照ください。

6.1 統合開発環境（IDE）

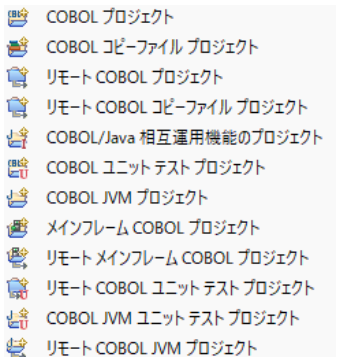
業界標準の IDE である Eclipse、Visual Studio、Visual Studio Code を利用して、コーディング、コンパイル、デバッグ、テストを効率的に行うことができます。また、テキストエディターでソースを編集後、コマンドを利用したコンパイルもでき、開発者に合わせた開発スタイルを選択できます。

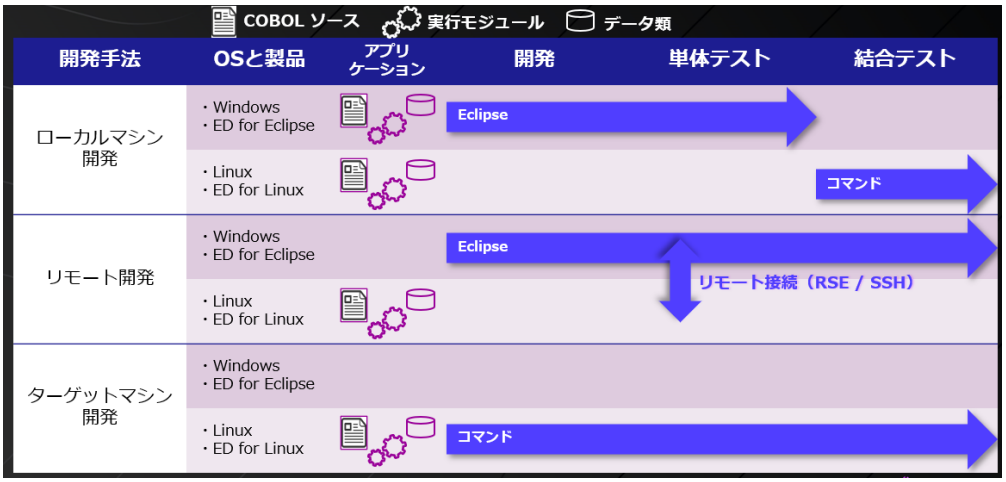


IDE ではデバッガを利用することにより、ステップ実行しながら変数の値を確認するなど、開発業務の効率化や品質を担保することができます。

6.2 COBOL プロジェクト

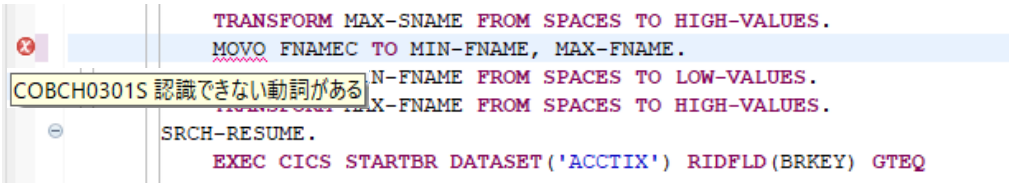
Eclipse と Visual Studio では COBOL 専用のプロジェクトを作成して、関連する COBOL ソースや JCL などを配置します。また、Eclipse から Linux マシンに接続し、Linux マシンに配備したソースを編集後、実行モジュールを Linux マシンに生成できるリモートプロジェクトも利用できます。





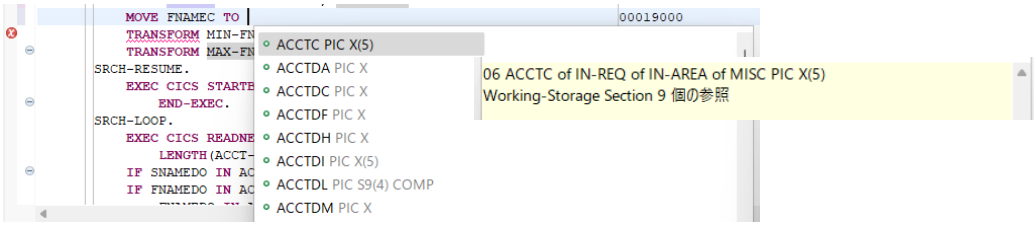
6.3 COBOL エディタ機能

COBOL プロジェクトに配置されたソース類は、COBOL 専用エディタを使用してコーディングを行います。コーディング時に即時エラー判定が可能なバックグランドパーシング機能を利用することができます。

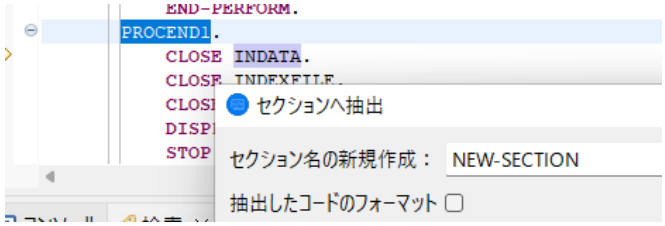


また、項目定義の位置へジャンプすることや、マウスオーバーによるデータ属性の確認、転送先候補の表示ができ、コーディングミスを未然に防ぐことで、開発工数の削減に貢献します。また、ソースコードのリファクタリングも可能です。

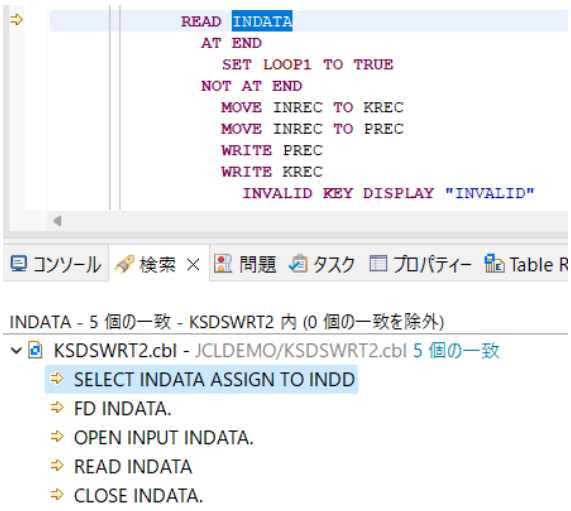
転送先候補を提供するコンテンツアシスト例）



ソースのリファクタリング例）

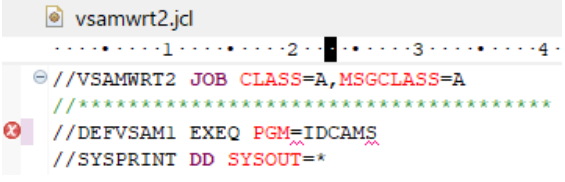


参照箇所の検索例）



6.4 JCL エディタ機能

JCL 専用のエディタを利用してコーディングを行うことができます。COBOL ソースと同様に、即時エラー判定が可能なバックグランドパーシング機能を利用することができます。



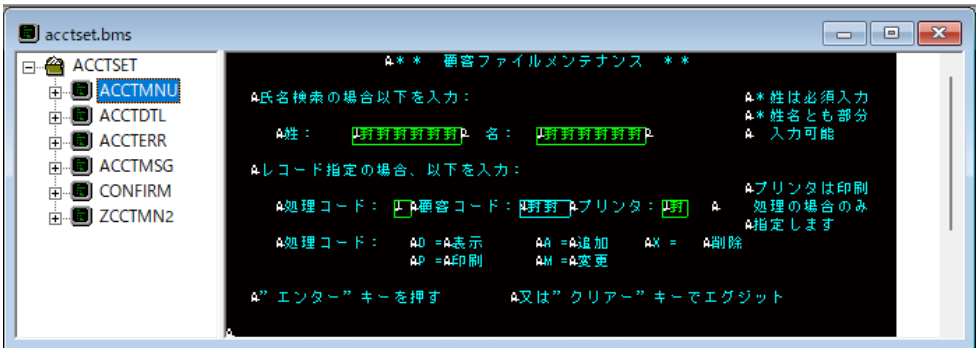
6.5 CICS 構文のサポート

EXEC CICS 構文や API、SPI コマンドの大部分をサポートしています。BMS 画面定義もサポートしており、画面イメージでメンテナンス可能な BMS ペインターや、Eclipse の BMS プレビューを利用できます。

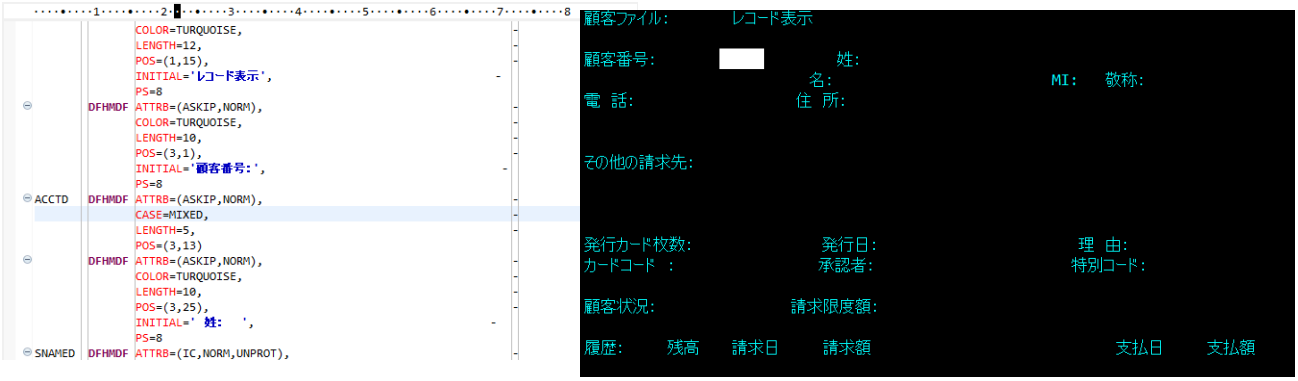
COBOL ソースコード例）

```
EXEC CICS SEND
          MAP('ACCTMNU')
          MAPSET('ACCTSET') FREEKB
          ERASE MAPONLY
END-EXEC
EXEC CICS RETURN TRANSID('AC01') END-EXEC
```

BMS ペインター例）



Eclipse の BMS プレビュー例）



6.6 IMS 構文のサポート

CALL インターフェイスである CBLTDLI や EXEC DLI といった IMS プログラム構文や MFS 定義をサポートしており、データのセグメント構成も保持できます。また、IBM メインフレームで使用している IMS 資源定義マクロをそのまま利用することができるため、新しくマクロを作成する必要がありません。JCL で使用する MPP、BMP、DLI もサポートしています。

COBOL ソースコード例）

```
PROCEDURE DIVISION   USING PCB-LT
                           ALT-PCB
                           DEMO-PCB.
    ENTRY 'DLITCBL'  USING PCB-LT
                           ALT-PCB
                           DEMO-PCB.
```

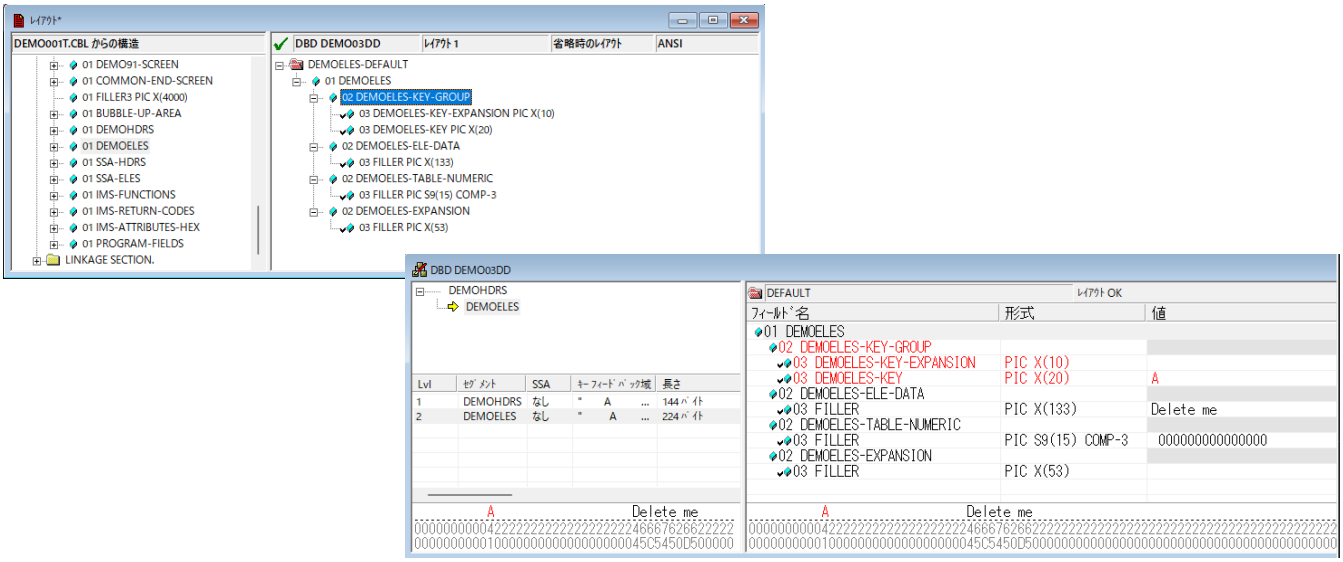
MFS 定義例）

```
OTDEMO91 MSG   TYPE=OUTPUT,SOR=(DEMO91,IGNORE),FILL=NULL,PAGE=YES,
               NXT=INDEMO91
         SEG
         MFLD  FLD0000,LTH=0079
         MFLD  FLD0001,LTH=0008
         MFLD  FLD0002,LTH=0035
         MFLD  LTERM,LTH=0008
```

JCL 例）

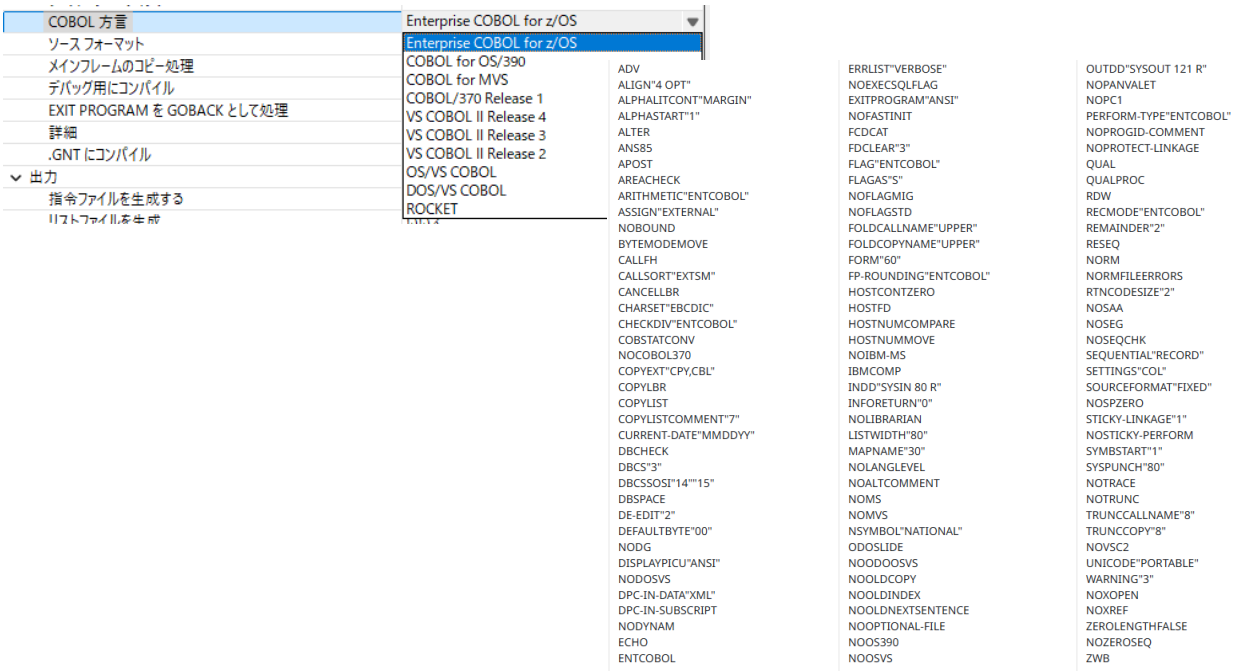
```
//S01       EXEC PGM=DFSRRC00,REGION=4M,
//            PARM='BMP,DEMO001B,DEMO001T,,,,,,,,CDLI,,N,N'
```

また、IMS データのセグメントレイアウトに沿ってデータのメンテナンスを行うことができるデータファイルツールも付属されており、レコードレイアウトを利用してデータを管理することができます。



6.7 COBOL コンパイラ指令

COBOL プロジェクトまたはソースファイルにプロパティとしてコンパイラ指令を指定します。Enterprise Developer は IBM メインフレームの COBOL コンパイラバージョンに沿った方言を指定でき、これにより挙動を模倣するよう、複数のコンパイラ指令が暗黙的に指定され、高い互換性を保つことができます。



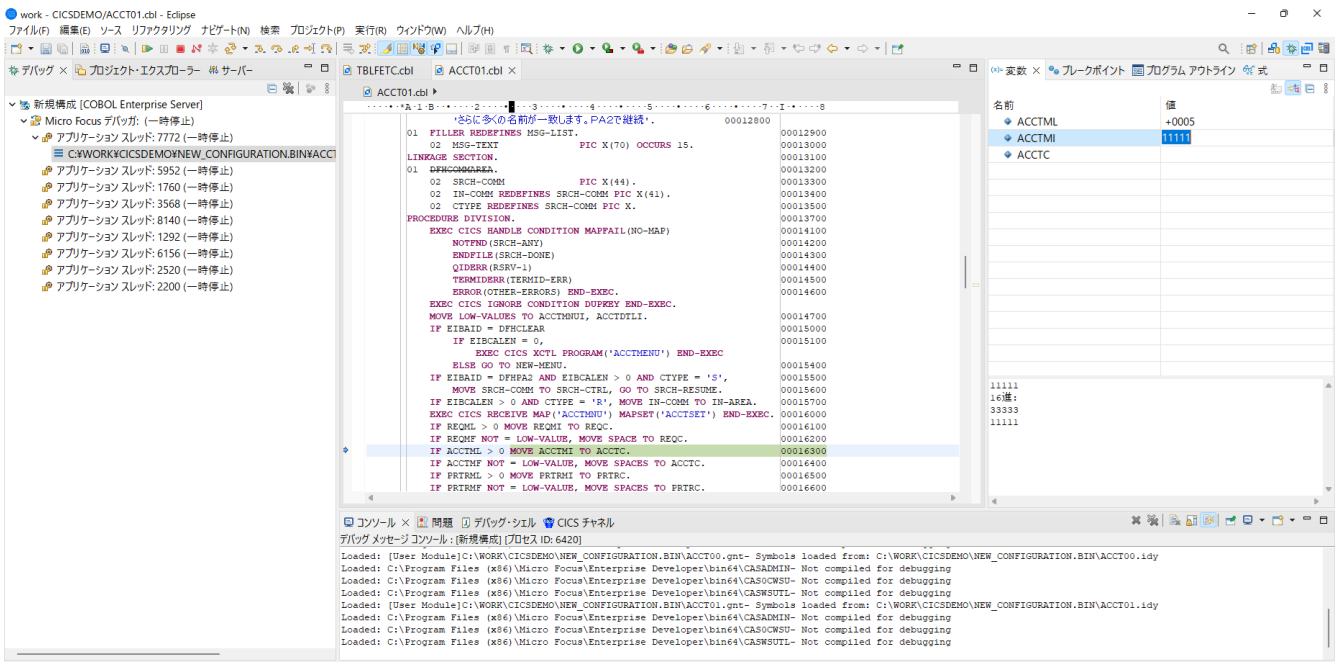
注意）

IBM メインフレームで指定しているコンパイラ指令を精査し、デフォルトと異なる指令が存在する場合は同様の意味を持つコンパイラ指令を追加指令として指定します。その他の注意点については、「IBM メインフレームコンパイラとの違い」の章で説明します。

6.8 プログラムデバッグ機能

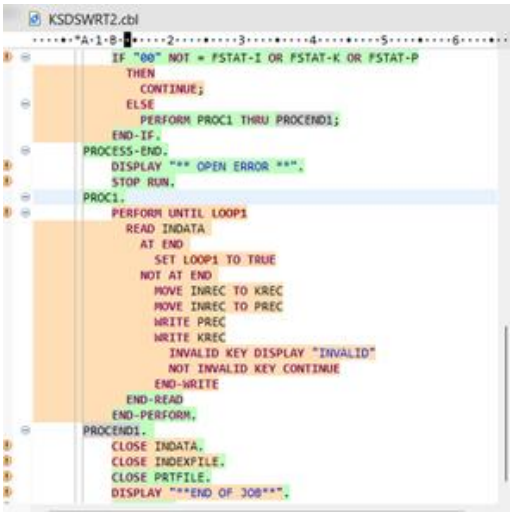
COBOL ソースのデバッグは、各 IDE 内部のデバッガを利用したプログラムのステップ実行ができます。変数の値を 16 進数で確認することや、動的な通過ルートの確認ができ、単体テストの品質向上に貢献できる機能です。

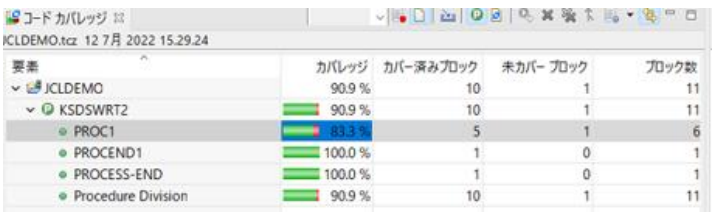
Eclipse のデバッグ画面例）



6.9 プログラムカバレッジ機能

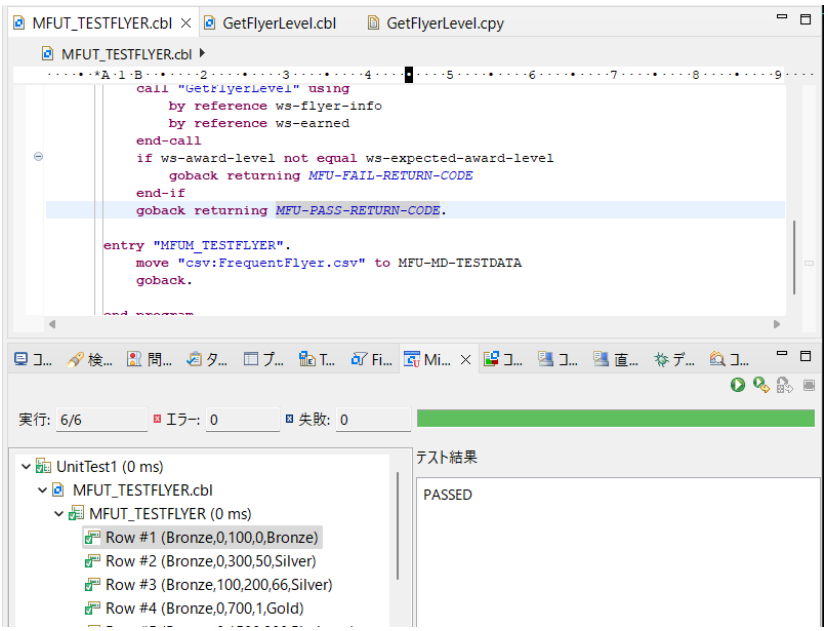
デバッグ機能に加えてカバレッジ機能を利用すれば、通過ロジックは緑、未通過ロジックはオレンジと色分けされたカバレッジ結果が出力され、静的にも通過ルートが確認でき、より品質を高めることができます。





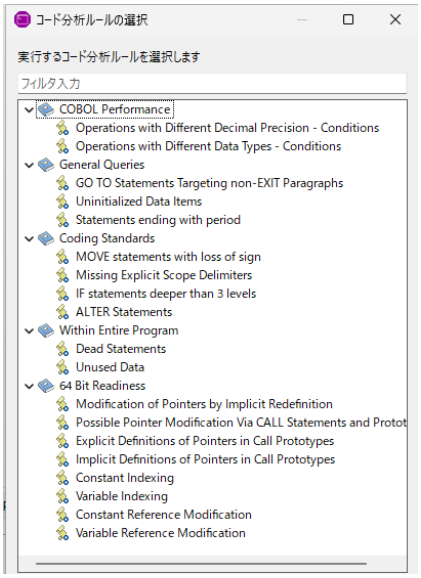
6.10 単体テストフレームワーク

コマンドラインまたは IDE から利用できるテストフレームワークです。COBOL のテスト仕様書を作成するイメージでテスト内容をコーディングし、結果を返却します。CI ツールと連携してテストの合否を判定できます。



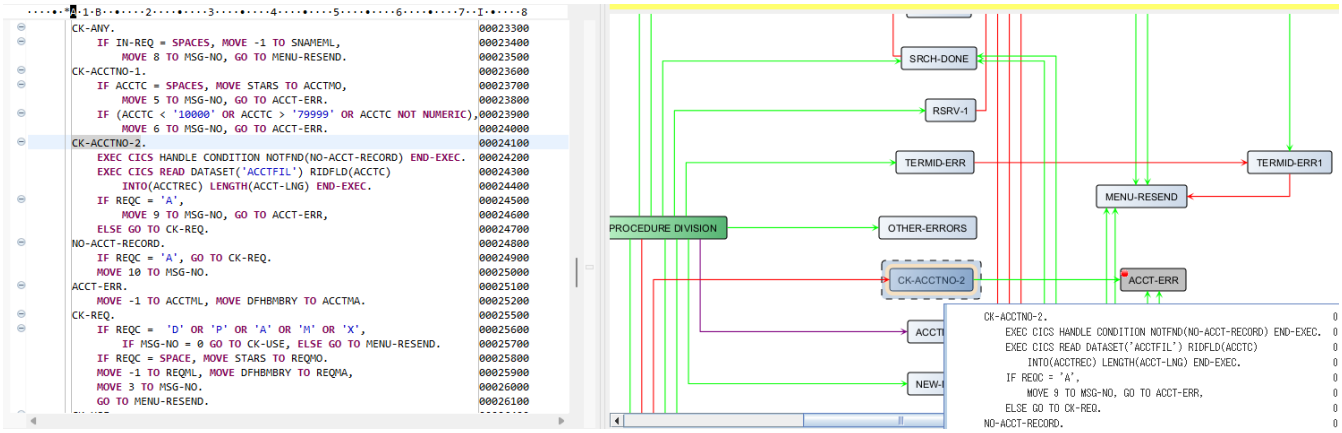
6.11 COBOL コードの解析

標準的な静的解析ルールが用意されており、コンパイル後など、
任意のタイミングで実施することができます。
これにより、通過しないルートの発見や、ルールに沿ったコーディングを
支援します。また、Enterprise Analyzer で作成した
独自のルールをインポートすることもできます。



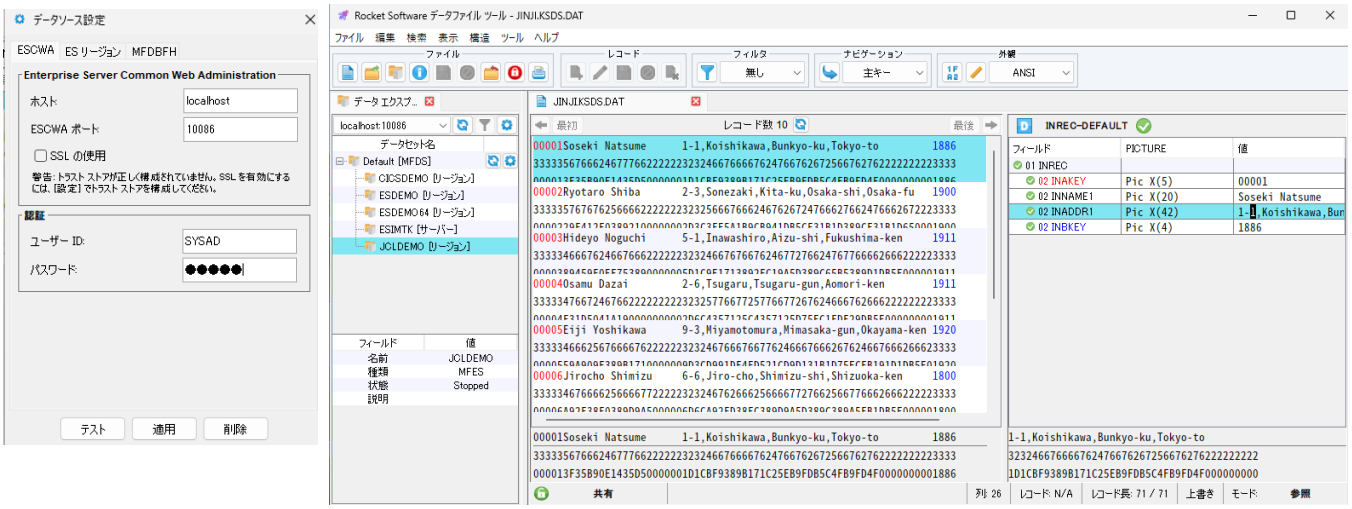
6.12 COBOL プログラムのフローグラフ

複雑なプログラムの分析や可視化をサポートする機能です。フロー描画のセクションや矢印をダブルクリックすると、ソースコードと連動して該当コードに位置づけられます。また、マウスオーバーにより該当コードをツールチップで確認することもできます。



6.13 データファイルツール

EBCDIC 文字コード、ASCII/SJIS 文字コートを持つデータをメンテナンスできるツールです。プログラムから選択したレコードレイアウトに沿ったデータメンテナンスや、COMP-3 などの COBOL 特有のデータ型を数値として扱うことや、16 進数を扱うこともできます。また、Enterprise Server インスタンスの JES カタログファイルと連携したデータメンテナンスを行うことができます。



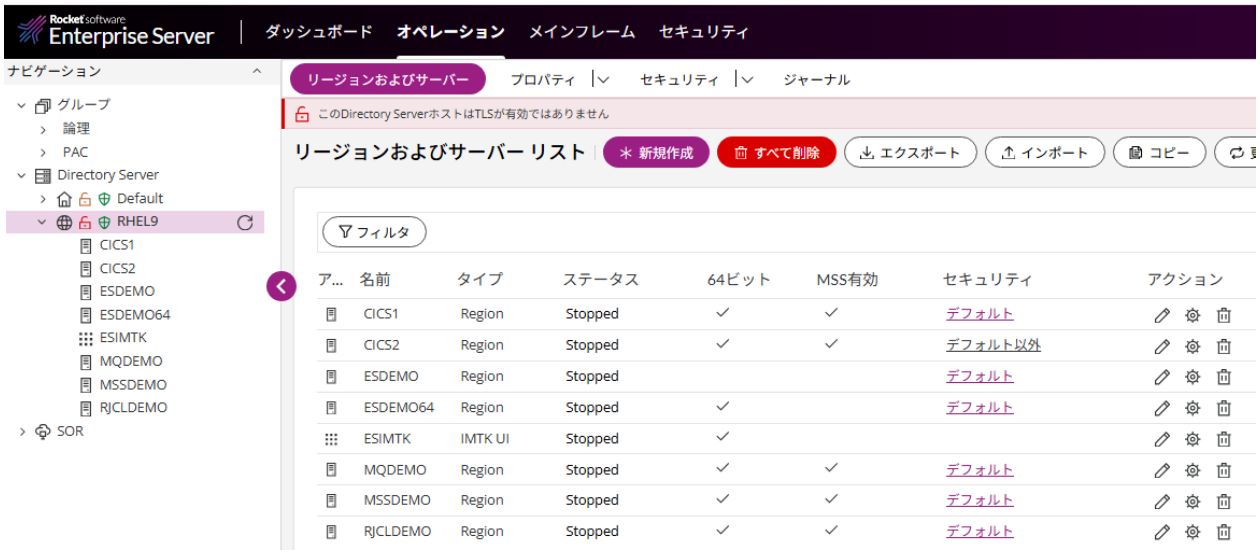
7. Enterprise Server を利用した実行

Enterprise Server には利便性を追求した機能や、IBM メインフレームとの互換性を保つための機能、運用と管理をサポートする機能など、様々な機能が備わっています。この章では実行環境の機能について説明します。

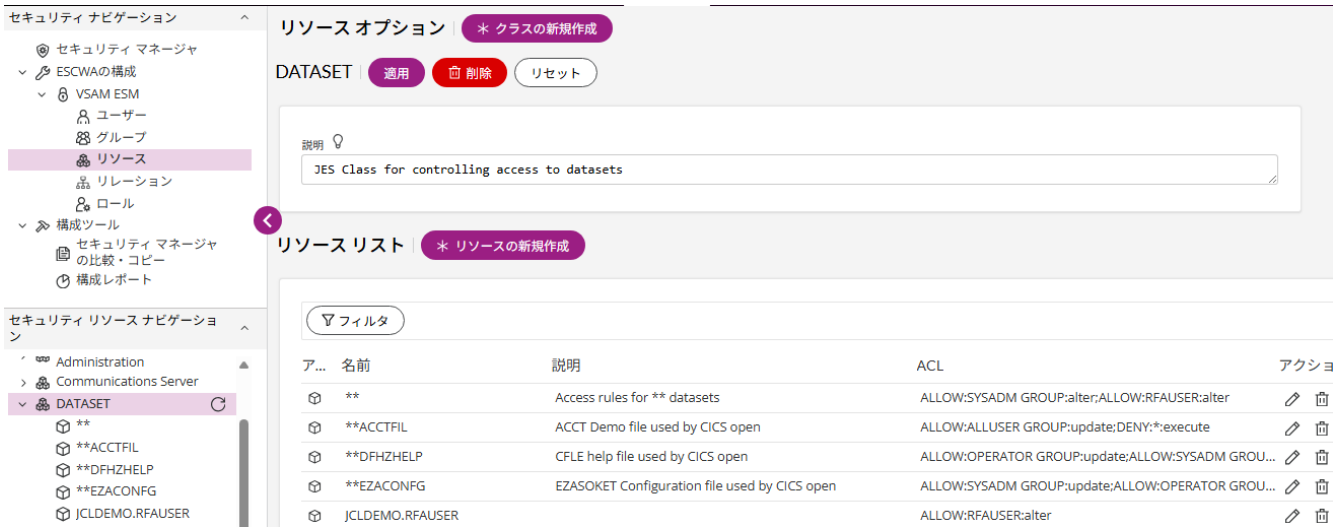
7.1 Enterprise Server Common Web Administration（略称 ESCWA）

実行結果など、運用管理者が参照する Enterprise Server インスタンスの管理画面です。Web ベースの画面を利用することにより、物理的に異なり、かつ異なる OS を持つホストマシンに作成した Enterprise Server インスタンスを一括管理することができます。

Windows と Linux マシンの Enterprise Server インスタンスの管理例）



また、製品に付随しているデフォルトの VSAM ESM セキュリティを使用すれば、ユーザー、グループ単位での細やかなアクセス制限を利用できます。



7.2 JES 機能

個々の Enterprise Server インスタンスが持つ JES 互換機能により、IBM メインフレームで使用していた JCL をオープン環境でも実行できます。一般的に使用される IBM JCL ユーティリティの多くをサポートしており、スプール、カタログファイル、プロシージャ、世代管理ファイルもサポートしています。また、ACCEPT 文による入力要求に対する返答も可能です。

JES スプール画面例）

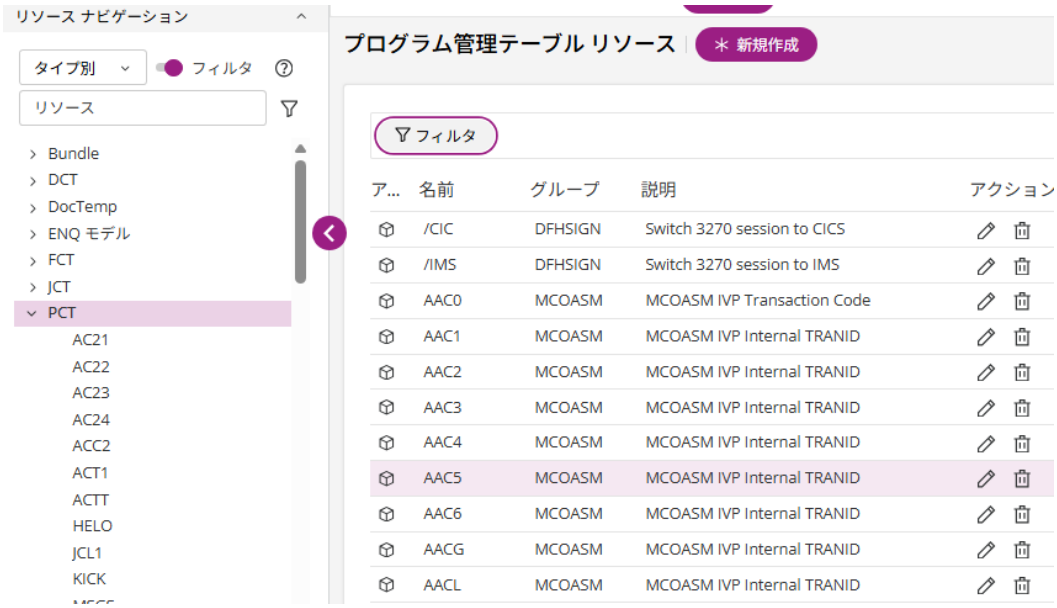
JES カタログファイル DCB 情報例）

JES カタログプロシージャ情報例）

7.3 CICS 機能

個々の Enterprise Server インスタンスが TP モニターの役割を持つため、OLTP ツールを別途用意する必要はありません。TN3270 エミュレータを Enterprise Server インスタンスが持つリスナーポートへ接続するだけで BMS ファイルに定義された画面が表示されます。また、CICS リソース定義ファイルに登録された PCT や FCT などを使用したアプリケーションの実行が可能です。

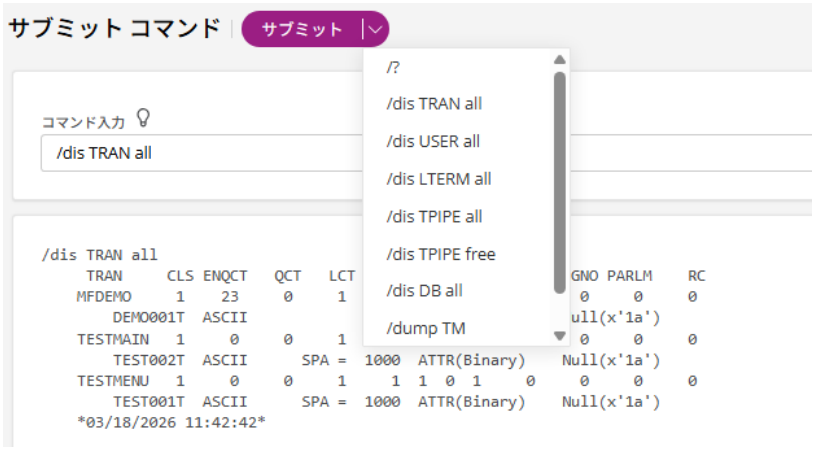
CICS リソース管理画面例）



7.4 IMS 機能

IMS オンライン処理においても、CICS 機能と同様に Enterprise Server インスタンスが TP モニターの役割を持つため、OLTP ツールを別途用意する必要はありません。TN3270 エミュレータを Enterprise Server インスタンスが持つリスナーポートへ接続するだけで MFS ファイルに定義された画面が表示され、IMS トランザクションを実行できます。また、IMS データベース制御コマンドを利用することもできます。

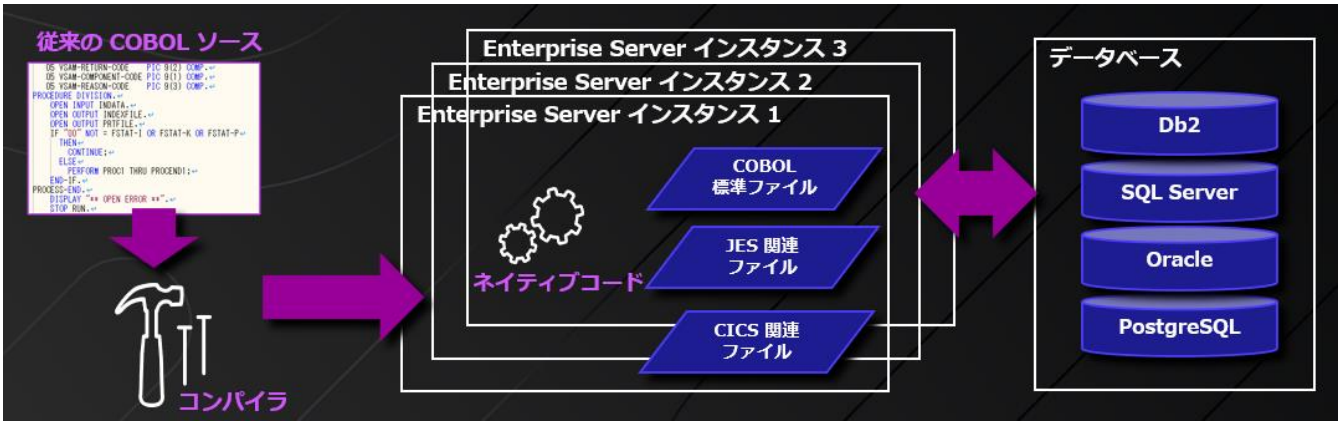
IMS コントロール画面例）



IMS トランザクション例）

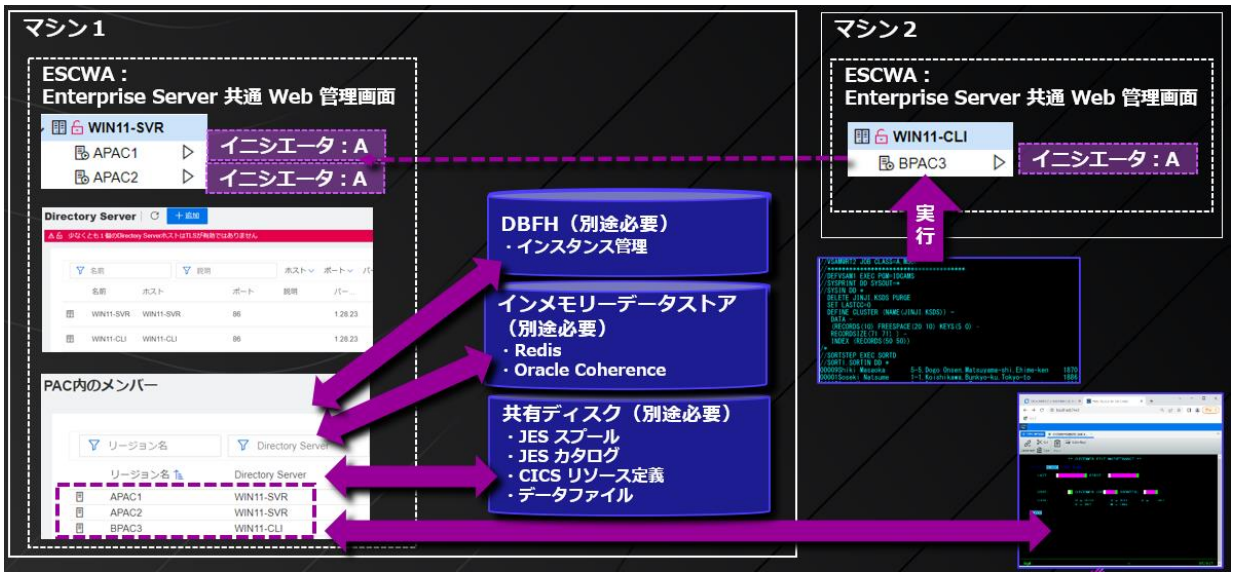
7.5 Database File Handler(DBFH)機能

ソース記述の変更なく、JCL、CICS で使用するデータファイルをデータベースに格納できる機能です。これを利用することで、JES 機能で使用するカタログファイルや CICS の FCT に登録されたファイルなどを、複数の Enterprise Server インスタンス間で共有することができます。レコードレイアウトをデータファイルと関連付けることにより、SQL 文の SELECT ステートメントからレコードを参照することもできます。ただし、データの格納形式は一般のテーブル項目属性とは異なります。



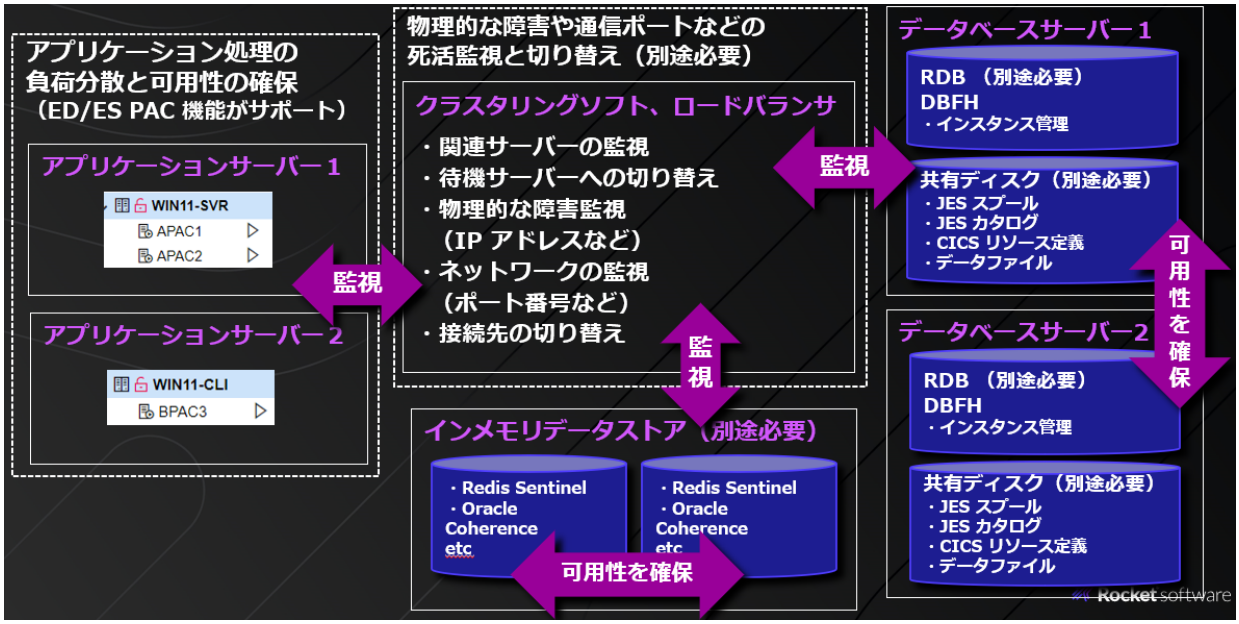
7.6 スケールアウト パフォーマンス/可用性クラスター機能(PAC)機能

複数の Enterprise Server インスタンスをグループ化し、JCL の負荷分散を実現することができます。OS、製品バージョンなどの前提条件が一致すれば、異なるホストマシンに存在する Enterprise Server インスタンスをグループに含めることができ、物理的なリスクも回避した運用を実現することができます。また、連携をとるために、RDB とインメモリーデータストアを使用します。



ただし、製品機能のみでシステム全体の可用性を確保できるわけではありません。RDB やインメモリーデータストアの可用性、使用ポートの死活監視などの周辺ツールが必要になります。

システム構成の例）



8. IBM メインフレーム コンパイラとの違い

IBM メインフレームと Enterprise Developer ではコンパイラが異なるため、これに起因して移行した COBOL ソースがコンパイルエラーとなる可能性があります。次に示す代表的なエラーが発生し、かつ Enterprise Developer のサポートがない場合は独自のプリコンパイラを使用する、もしくはプログラムを修正するなどの対策が必要となります。

8.1 COBOL 構文の差異

① 準拠する COBOL 規格が異なる場合

Enterprise Developer のサポート内容:

コンパイラ指令により EXAMINE などの旧構文を使用可能にします。

② 古い COBOL コンパイラ方言を使用している場合

③ COBOL 言語仕様としては不正であるが IBM メインフレームコンパイラが許容している場合

④ 追加された予約語が利用者語と衝突している場合

Enterprise Developer のサポート内容:

REMOVE コンパイラ指令により予約語から排除することができます。また、別の予約語に読み替える指令も用意しています。

⑤ 特殊構文を使用している場合

Enterprise Developer のサポート内容:

コンパイラ指令により PANVALET ++INCLUDE 文などを使用可能にします。

8.2 COBOL 実行時挙動の差異

COBOL 標準言語仕様により結果不定と定義されている事柄について、IBM メインフレーム実行時と Enterprise Server インスタンス実行時の挙動が異なる可能性がある代表的なものを列挙します。

① ON SIZE ERROR 句が指定されていない桁あふれの結果

Enterprise Developer のサポート内容:

CHECKDIV, HOSTARITHMETIC などのコンパイラ指令により不定の結果を回避できます。

② 数値データ項目に数値以外の値が含まれている場合

③ 初期値を設定していないデータを参照する場合

Enterprise Developer のサポート内容:

INIT-BY-TYPE、DEFAULTBYTE コンパイラ指令により初期値を設定することが可能です。

④ 算術結果が異なる場合

Enterprise Developer のサポート有:

Enterprise Developer では IBM 社が公開している中間結果精度の仕様に互換性を持つコンパイラ指令を用意しています。例えばコンパイラ指令 ARITHMETIC”OSVS”を指定した場合は、OS/VS の規則に則って切り捨てを行うなど、COBOL バージョンの指定に依存して中間結果仕様を適用しています。

適用する動作：

ENTCOBOL

Enterprise COBOL for z/OS および OS/390 の規則に従って切り捨てを行います。

MF、ISO2002、ANSI

中間結果を切り捨てません。ARITHMETIC"MF"、ARITHMETIC"ISO2002"、または ARITHMETIC"ANSI" を指定した場合、切り捨てが行われないため、式の計算が最も正確になります。

OS390

COBOL for OS/390 の規則に従って切り捨てを行います。

OSVS

OS/VS COBOL の規則に従って切り捨てを行います。

VSC2

VS COBOL II および COBOL/370 の規則に従って切り捨てを行います。

9. Enterprise Developer チュートリアルと例題

Enterprise Developer の製品マニュアルには COBOL と PL/I ユーザーに向けた JCL、CICS、IMS などの様々なチュートリアルを準備しています。これらのチュートリアルには IDE を使用したプロジェクトの作成方法からコンパイル、実行、デバッグまでを具体的に記述しており、例題もダウンロードできます。

メインフレーム チュートリアル

このセクションには、メインフレーム上での開発を対象としたチュートリアル (CICS、IMS、JCL、および Open PL/I アプリケーションの開発方法など) が含まれています。

Enterprise Developer - CICS チュートリアル
このチュートリアルのセットは、Enterprise Developer for Eclipse による CICS アプリケーションの開発および移行方法を案内します。
使用する例題はこちらからダウンロードしてください。

Enterprise Developer - JCL チュートリアル
このチュートリアルのセットは、Enterprise Developer for Eclipse による JCL バッチアプリケーションの開発および移行方法を案内します。
使用する例題はこちらからダウンロードしてください。

Enterprise Developer - IMS チュートリアル
このチュートリアルのセットは、Enterprise Developer for Eclipse による IMS アプリケーションの開発および移行方法を案内します。
使用する例題はこちらからダウンロードしてください。

Enterprise Developer - DB アクセスチュートリアル
このチュートリアルのセットは、Enterprise Developer for Eclipse による SQL アクセスの開発および移行方法を案内します。
使用する例題はこちらからダウンロードしてください。

Enterprise Developer - PL/I JCL チュートリアル
このチュートリアルでは Enterprise Developer for Eclipse による PL/I 言語の JCL アプリケーション開発および移行方法を案内します。

10. おわりに

IBM メインフレーム環境では資源を開発者全員で共有するために、急を要するコンパイルは優先度の調整を行う、変数の値を確認するためにデバッグ文を入れたプログラムを再コンパイルする、などの開発スタイルが多く見受けられます。一方、オープン環境の Enterprise Developer は Eclipse, Visual Studio といった業界標準の IDE にアドオンする形で COBOL、PL/I アプリケーションの開発ができ、Visual Studio Code では COBOL 拡張機能をインストールすることで COBOL、PL/I の開発作業をより軽く迅速に行うことができます。

ステップ実行を利用したデバッグ、変数値の動的な確認、処理の流れの動的な把握など、他開発言語と同様の開発スタイルで、効率的な開発環境を整えることができる製品です。
また、開発用実行環境の Enterprise Server インスタンスを開発者ごとに占有することができるため、コンパイルや実行時に他開発者との調整は必要なく、これによる開発工数の削減も見込めます。

Enterprise Developer の機能をご理解いただき、開発業務のモダナイゼーションをご検討いただければ幸いです。

補足:稼働環境

本書は下記環境を基に記述しています。

1）OS

Windows 11 Pro

2）プロセッサ

13th Gen Intel® Core(TM) i7-13800H 2.92 GHz

3）システムの種類

64 ビットオペレーティング システム x64 ベース プロセッサ

4）Rocket 製品 バージョン

Enterprise Developer 11.0J

注意)Enterprise Server 11.0J と同等の開発用実行環境を含んでいます。

5）使用した製品マニュアル バージョン

Enterprise Developer 11.0 for Eclipse