佐田 雄一

株式会社ミガロ.

RAD事業部 技術支援課

[Delphi/400] Delphi 10シリーズ
VCLプログラム開発の最新トピックス

略歴
1985 年 12 月 6 日生まれ
2009 年 3 月 甲南大学 経営学部卒業
2009 年 4 月 株式会社ミガロ. 入社
2009 年 4 月 システム事業部配属
2019 年 4 月 RAD 事業部配属

現在の仕事内容
Delphi/400 を利用したシステム開発や保守作業の経験を経て、現在は Delphi/400 のサポート業務を担当。

1. はじめに

2020 年 5 月に、Delphi の最新バージョンとなる「10.4 Sydney」がリリースされた。

Delphi および Delphi/400 では、コンポーネントをフォームに配置してプロパティを定義し、プログラムをコーディングしていく、「ビジュアルプログラミング」と呼ばれる開発手法でアプリケーションを作成する。本稿では、2015 年に登場した Delphi 10 シリーズの初版である「Delphi 10 Seattle」から、最新の「Delphi 10.4 Sydney」の間で進化したビジュアルプログラミングの手法を紹介する。

なお、本稿執筆（2020 年 9 月）時点で Delphi は「10.4 Sydney」が最新だが、IBM i に接続するためのミドルウェア「Delphi/400」は「10.2 Tokyo」が最新となっているため、今後のリリースにご期待いただきたい。

また、「Delphi 10 Seattle」より前の新機能については 2016 年発行のミガロ. テクニカルレポートにある『Delphi/400 最新プログラム文法の活用法』を参照いただきたい。

2. 最新Delphi活用 TIPS

本章では、10 Seattle 以降の新しい Delphi ならびに Delphi/400 を使って開発効率をさらに向上するための TIPS をいくつか紹介する。

なお、10 Seattle で Delphi/400 に正式対応となった「FireDAC 接続」、10 Seattle で追加された IoT や Beacon コンポーネントを使った開発テクニック、10.2 Tokyo で追加された各種クラウドサービスに接続する「Enterprise Connectors」については、それぞれ過去のテクニカルレポートで紹介しているのでそちらを参照いただきたい。

2-1. GetIt パッケージマネージャ（10 Seattle ～）

Delphi の統合開発環境（IDE）の 10 Seattle 以降では、「GetIt パッケージマネージャ」を使用して、Delphi の開発元であるエンバカデロ社が提供する追加コンポーネントや、サードパーティ製品の評価版などを簡単にインストール／アンインストールできる。

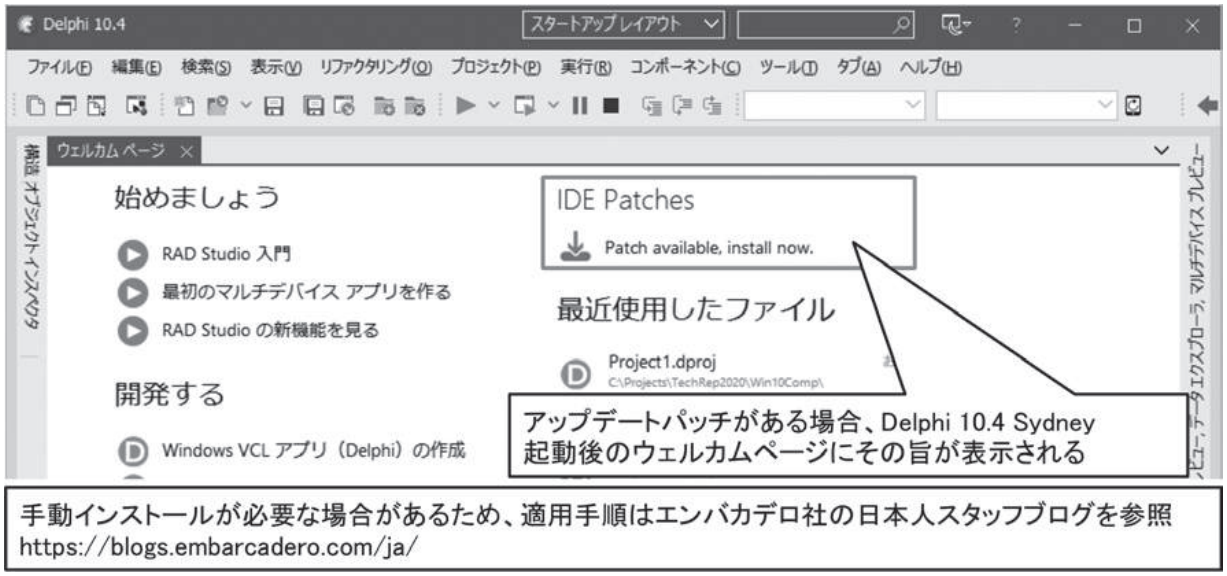
「ツール」メニューに追加された「GetIt パッケージマネージャ」を選択すると、【図 1】のような画面が起動するので、ここに表示されている一覧から目的の項目をインストール／アンインストールする。Delphi のバージョンによって、画面のレイアウトや提供ツールのラインナップは若干異なっている。

また Delphi 10.4 Sydney では、エンバカデロ社が製品の修正パッチを発行した際に、この GetIt パッケージマネージャでもダウンロード可能になっている。

この修正パッチはダウンロードしただけでは適用されずに手動インストールを必要とする場合がある。手元の Delphi 10.4 Sydney を起動した際に、ウェルカムページに【図 2】のような画

図1　GetItパッケージマネージャ

図2　製品アップデートパッチの確認



手動インストールが必要な場合があるため、適用手順はエンバカデロ社の日本人スタッフブログを参照
https://blogs.embarcadero.com/ja/

ソース1

インライン変数宣言①

```
// Delphi 10.4 Sydneyより前は、
// コード開始前のvarブロックで宣言する必要があった
procedure Test1;
var
  I, J: Integer;
begin
  I := 24;
  J := 26;
  ShowMessage(IntToStr(I + J));
end;

//---------------------------------------------------------------------------

// Delphi 10.4 Sydneyでは、
// コード内で直接変数を宣言できるようになった
procedure Test2;
begin
  var I: Integer := 24;
  var J: Integer;
  J := 26;
  ShowMessage(IntToStr(I + J));
end;
```

構文の途中でvar宣言や初期化が可能

面が表示された場合は、エンバカデロ社のホームページにある日本人スタッフブログを参照いただきたい。

2-2. インライン変数宣言（10.4 Sydney）

従来の Delphi のコーディングでは、伝統的な Pascal 言語の規約に従い、すべての変数は関数、プロシージャ、メソッドのコードの開始前に var ブロックで定義する必要があった。

しかし、さらなる柔軟性を提供する新しいインライン変数宣言のコーディングが可能となり、コードブロック内で直接変数を宣言できるようになった【ソース 1】。ここで宣言された変数はそのコードブロック内（「begin ～ end;」の中）でのみ有効で、その外側から参照しようとするとコンパイルエラーになる。【ソース 2】

コードブロック内でのインライン変数宣言がとくに有効となるのが、変数宣言を伴う for ループで、「for var I: Integer := 1 to 10 do ……」といった記述方法が可能である。10.4 Sydney でこのように記述した場合、ループの外側でループ変数を参照しようとするとコンパイルエラーになるので、意図しないループ変数の参照も防げる。

2-3. Language Server Protocol（10.4 Sydney）

Delphi の開発者であれば、IDE でコードを入力しているときに Ctrl ＋スペースキーを押してコード補完を使用したことがあるだろう。

このコード補完を表示する手法をコードインサイトと呼ぶ。Delphi 10.4 Sydney のコードインサイトでは、Language Server Protocol（以下、LSP）が採用されており、LSP はこのコード補完の結果を Delphi 自身ではなく別プロセスで計算する手法をとっている。

従来のバージョンでは Delphi 自身がコードインサイトを行っていたため、Ctrl ＋スペースキーを押してからコード補完が表示されるまでに、コンマ数秒～数十秒の待機時間が必要であった。

しかし Delphi 10.4 Sydney では、待機時間なしでコード補完が可能である（待機時間の間も Delphi を操作できる）。それに加えて、内部フィルタリングロジックの変更によって従来の前方一致だけでなく中間一致にも対応しており、プロパティやメソッドがより見つけやすくなっている。【図 3】

また、別プロセスでコードインサイトを行うもう 1 つの利点として、従来のバージョンでは実行できなかったデバッグ中のコード補完の表示も可能となる点が挙げられる。【図 4】

2-4. プロジェクトオプションの設定項目充実（10 Seattle ～）

IDE の「プロジェクト」メニューにある「オプション」を選択すると、開いているプロジェクトのコンパイル先、アイコン、バージョン情報といったさまざまな設定が可能である。10 Seattle 以降は、このプロジェクトオプションも目覚ましい進化を遂げている。【図 5】

＜高 DPI の認識＞

高 DPI は高精細液晶や高解像度液晶とも呼ばれており、Windows 10 のディスプレイ設定にある「拡大縮小とレイアウト」の「テキスト、アプリ、その他の項目のサイズを変更する」の項目を指す。

対応しているアプリケーションを起動すると、実行 PC のディスプレイの DPI の設定が 100％でない場合にフォームやフォーム内の各項目のサイズが自動拡張され、解像度の高いディスプレイでも画面が小さくなりすぎずに表示できる。【図 6】

Delphi では、10 Seattle からこの高 DPI 設定に対応しており、プロジェクトオプションの「アプリケーション」の項目から、マニフェストの設定を変更することで指定が可能となっている。バージョンごとに設定できる項目や項目名が少しずつ異なるので、バージョン間の差異については【図 7】に記載する。

＜管理者権限の有効化＞

コンパイルしたアプリケーションをダブルクリックなどで実行する場合に、実行時の権限を指定できる。

プロジェクトオプションの「アプリケーション」の項目から、前述の高 DPI と同様に、マニフェストの設定にある実行レベルを変更することで指定できる。【図 5】

デフォルトの「インボーカとして」を選択している場合は通常権限（親プロセスと同じ権限）で実行し、「使用可（最高）」を選択している場合はユーザーが取得できる最も高い権限で実行する。

そして「管理者権限が必要」を選択している場合は、アプリケーションを強制的に「管理者として実行」で実行させる。こちらも前述の高 DPI と同様、バージョンごとに設定できる項目や項目名が少しずつ異なるので、バージョン間の差異については【図 7】に記載する。

なお、「管理者権限が必要」の設定でコンパイルしたアプリケーションを Delphi 上でデバッグするには、Delphi 自身も管理者として起動しておく必要がある。【図 8】

＜カスタムスタイルの機能拡張＞（10.4 Sydney）

XE3 以降のバージョンでは、プロジェクトオプションでスタイルを設定することで、コンパイルされるアプリケーションの見た目を変更できる【図 9】。10.4 Sydney ではこのスタイル機能が進化し、同一アプリケーション内でのフォームやコントロール（Edit や Button など）単位でスタイルを変更可能になった。

プロジェクトオプションでチェックを入れたカスタムスタイルの中から、対象のフォームやコントロールの「StyleName」プロパティに、そのスタイル名を指定することで反映される。【図 10】

2-5. Windows 10 サポートと IDE の機能強化（10 Seattle ～）

Delphi 10 Seattle からは Windows 10 が正式にサポートされ、専用のユーザーインターフェースに対応したコンポーネントも追加されている。また、XE7 以前のバージョンと比較して IDE で利用可能なメモリが約 2 倍に拡張されており、大規模プログラムでも安定した IDE 環境で開発可能になっている。

加えて、非ビジュアルコンポーネントの表示切り替え機能が追加され、フォーム／ユニットの切り替えの右側に追加されたボタンを押すと、【図 11】のように Table や Query などの画面上に見えないコンポーネントの表示／非表示を切り替えられるようになった。これにより、画面項目の配置の微調整がさらに容易になっている。

ソース2

インライン変数宣言②

```
// コード内で直接宣言した変数は、そのbegin～end内でのみ有効
procedure Test3;
begin
  begin
    var I: Integer := 24;
    var J: Integer;
    J := 26;
  end;
  ShowMessage(IntToStr(I + J));
end;
```

これは変数宣言されたbegin～endの
外側にあるため、コンパイルエラーになる

図3　Language Server Protocol（LSP）

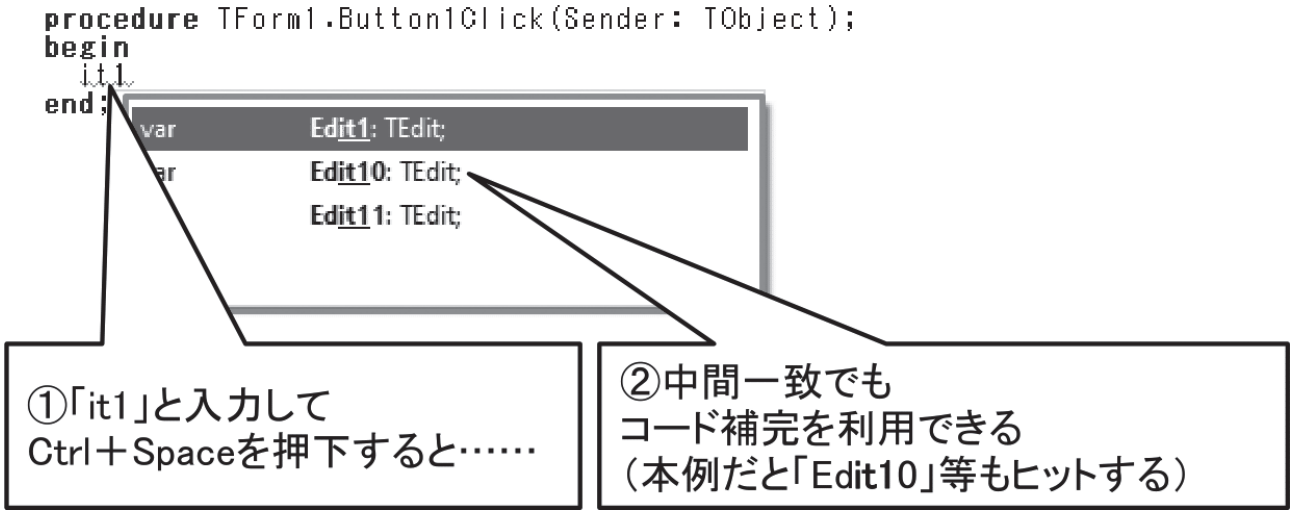
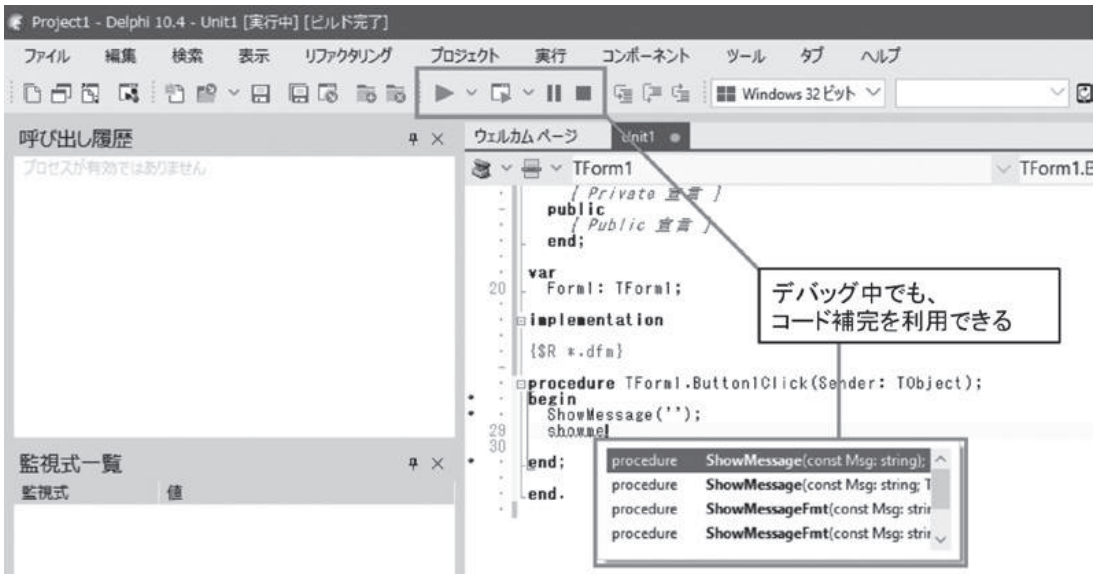


図4　Language Server Protocol（LSP）



2-6. Windows 10 ライクな新コンポーネント群（10 Seattle ～）

Windows 10 の正式サポートに伴い、その動作を模倣するコントロールがいくつか追加されている。これらのコンポーネント群はツールパレットの「Windows 10」パレットページに追加されているが、実際には Windows 10 ネイティブのコンポーネントではないので、古い Windows OS で実行しても動作できる。【図 12】

以下に、代表的なものをいくつか紹介する。【図 13】

・TSplitView

開いたり閉じたりできる他コントロールのコンテナで、FireMonkey の TMultiView と似た動作をする。Open メソッドおよび Close メソッドで、Windows 10 やスマートデバイスの画面のようにアニメーションでパネルが開閉する動作を再現できる。

・TSearchBox

TEdit の拡張で、Enter キーを押すか、虫眼鏡のボタンを押すことで「OnInvokeSearch」イベントが走り、検索画面への遷移といった処理を組み込める。

・TToggleSwitch

Windows 10 の設定画面などでよく見かける「オン」と「オフ」を切り替えるスイッチの動作を実現する。State プロパティを「tssOn」に設定すればオンに、「tssOff」に設定すればオフに切り替わる。

・TActivityIndicator

Windows 10 で各種処理中に表示されるインジケータ（ぐるぐる回るアイコン）を画面上の任意の位置に表示させる。

上記以外にも、Delphi 10.2 Tokyo 以降では Windows 10 ライクな日付時刻の表示や指定を行う TCalendarView ／ TDatePicker ／ TTimePicker ／ TCalendarPicker などが追加されている。

本項で紹介したコンポーネントはいずれも、【図 13】に記載した簡単なプロパティの設定や命令だけで扱えるので、ぜひ一度使い心地をご確認いただきたい。

2-7. TTitleBarPanel と CustomTitleBar（10.4 Sydney）

最新の Delphi 10.4 Sydney では、新しく追加されたコンポーネント「TTitleBarPanel」、および TForm に新たに追加されたプロパティ「CustomTitleBar」を利用することで、最新のオフィスソフトやブラウザのようにタイトルバーをカスタマイズできるようになった。

前項の Windows 10 ライクな各コンポーネントと比較すると少々複雑なので、使用手順を以下に解説する。

まず対象のフォームに TTitleBarPanel を配置し、タイトルバー内に表示させたい Button や Edit などを配置する。【図 14】

TTitleBarPanel の設定が完了したら、親となる TForm の設定を行う。CustomTitleBar プロパティを開き、「Control」に今作成した TTitleBarPanel を、「Enabled」に True を指定すると、フォームと TTitleBarPanel の紐付けが行われる。【図 15】【図 16】

アプリケーションをコンパイルして実行すると、【図 17】のような画面が表示され、タイトルバーがカスタマイズされていることが確認できる。

2-8. TEdgeBrowser（10.4 Sydney）

Delphi のフォーム上に TWebBrowser コンポーネントを配置すると、実行するアプリケーション内で Web ブラウザの画面を表示できる。しかし、従来この TWebBrowser 内で表示されるブラウザは IE パーソナリティ（Internet Explorer 7 ベース）として動作していた。

最新の Delphi 10.4 Sydney では新しく「TEdgeBrowser」コンポーネントが登場し、これを利用することで Edge パーソナリティによるブラウザを画面に表示できるようになった。【図 18】【ソース 3】

2020 年 9 月現在、Microsoft Edge は Windows の OS 標準コンポーネントになっていないため、TEdgeBrowser を動作させるには以下の 2 つのアイテムを PC にインストールする必要がある。

① Microsoft Edge Chromium ベースのブラウザ（Canary channel version）
② WebView2 コントロールの DLL

このうち①については、2020 年 9 月現在、通常の Edge の最新バージョンよりも新しい、Canary Channel Version（毎日更新されるデベロッパー向けビルド）の Edge を Microsoft のサイトからダウンロードし、インストールする必要がある。【図 19】

一方の②は、GetIt パッケージマネージャからインストールしたあと、【図 20】【図 21】のようなフォルダが開くので、対応 bit 数（コンパイルするアプリケーションが 32bit アプリケーションなら「win32」、64bit アプリケーションなら「win64」）のサブフォルダを開き、その中の「WebView2Loader.dll」を取得しておく。

IDE でアプリケーションを開発し、完成したアプリケーションと同じフォルダに「WebView2Loader.dll」をコピーすることで、【図 18】のように TEdgeBrowser でブラウザが画面に表示される。

なお、従来の TWebBrowser コンポーネントも機能拡張し、Edge パーソナリティに対応している。上記の TEdgeBrowser を動作させる①②のアイテムと「WebView2Loader.dll」が必要である点は変わらないが、その条件を満たしていれば、TWebBrowser コンポーネントに追加された SelectedEngine プロパティを、デフォルトの「IEOnly」から「EdgeOnly」に変更することで、Edge パーソナリティで TWebBrowser のブラウザを表示できる。【図 22】

また SelectedEngine プロパティを「EdgeIfAvailable」に変更すると、使用可能な場合のみ Edge パーソナリティを使用し、使用不可の場合は従来の IE パーソナリティでブラウザを表示する。どちらのブラウザが使用されているかは、TWebBrowser の ActiveEngine プロパティの値で確認できる。

注意点としては、TWebBrowser の各プロパティ・メソッド・イベントは従来の IE パーソナリティに準拠している点が挙げられる。Edge パーソナリティで使用している場合は一部のプロパティ・メソッド・イベントが互換してお

図5　Delphi 10.4 Sydneyのプロジェクトオプション

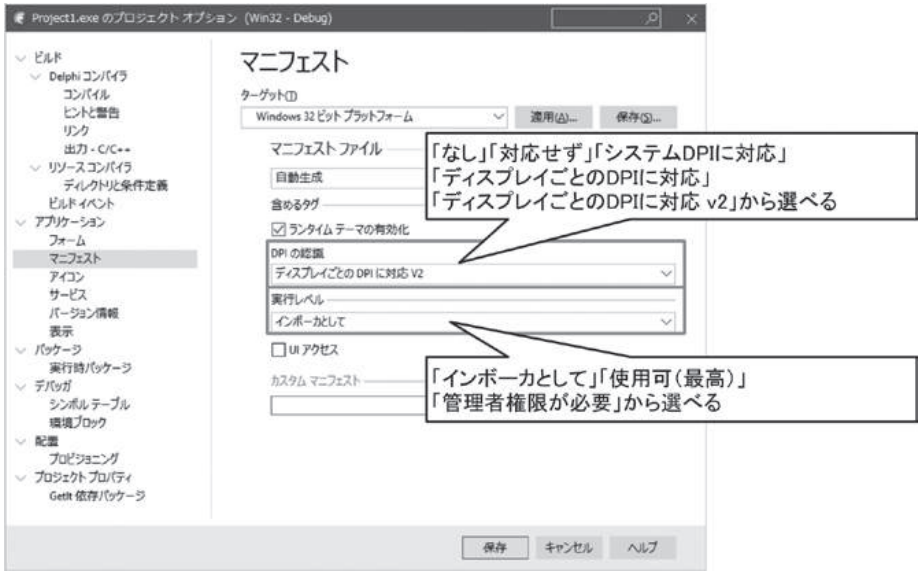


図6-1　高DPI設定対応

図6-2　高DPI設定対応

らず、プロパティの取得結果が変わったり、メソッドで例外が発生したりする場合がある。【図 23】

Delphi 10.4 Sydney を取得したあと、TWebBrowser から TEdgeBrowser への移行が可能な場合は、段階的に移行を検討いただきたい。

3. まとめ

株式会社ミガロ．では、開発元である仏 SystemObjects 社の間で日本総代理店契約を締結し、2000 年 9 月に Delphi/400 V5 の販売を開始してから今年で 20 周年を迎えた。

Delphi/400 は古いバージョンのまま互換の範囲で使い続けられることも魅力ではあるが、本稿で紹介した新機能を利用することで、より多彩な機能が活用できるようになる。

本稿の記述を参考に、最新バージョンの Delphi ならびに Delphi/400 に関心をもっていただければ幸いである。

M

図7　Delphi10シリーズの高DPI・実行レベルの設定

<高DPI対応の設定>

バージョン	XE7以前	10 Seattle	10.2 Tokyo	10.4 Sydney
プロジェクトオプション内の設定画面の場所	なし	アプリケーション → マニフェスト ファイル		アプリケーション → マニフェスト ファイル → DPI の認識
設定項目	設定不可	「高 DPIの有効化」チェックのオン／オフで切り替え（オンの場合「ディスプレイごとのDPIに対応」になる）		「なし」 「対応せず」 「システムDPIに対応」 「ディスプレイごとのDPIに対応」 「ディスプレイごとのDPIに対応 v2」から選択

※実行時の結果（実際にどのDPI設定になっているか）は、タスクマネージャの「詳細」タブから確認可能

<管理者権限の要求設定>

バージョン	XE7以前	10 Seattle	10.2 Tokyo	10.4 Sydney
プロジェクトオプション内の設定画面の場所	なし	アプリケーション → マニフェスト ファイル	アプリケーション → マニフェスト ファイル → 実行レベル	
設定項目	設定不可	「管理者権限の有効化」チェックのオン／オフで切り替え（オンの場合「管理者権限が必要」になる）	「インポーカとして」 「使用可（最高）」 「管理者権限が必要」から選択	

図8　管理者権限指定時の注意

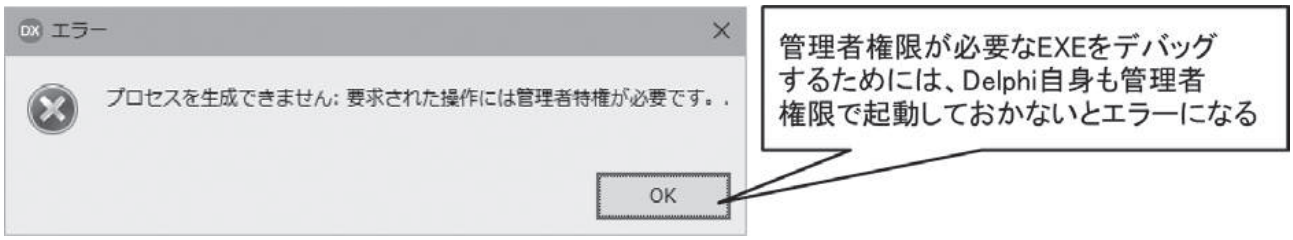


図9　カスタムスタイルの機能拡張

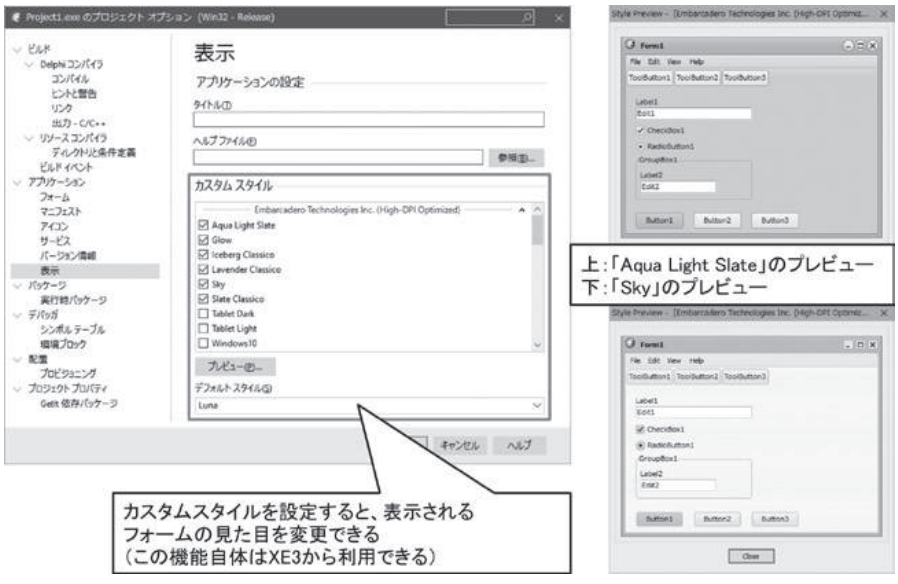


図10　カスタムスタイルの機能拡張

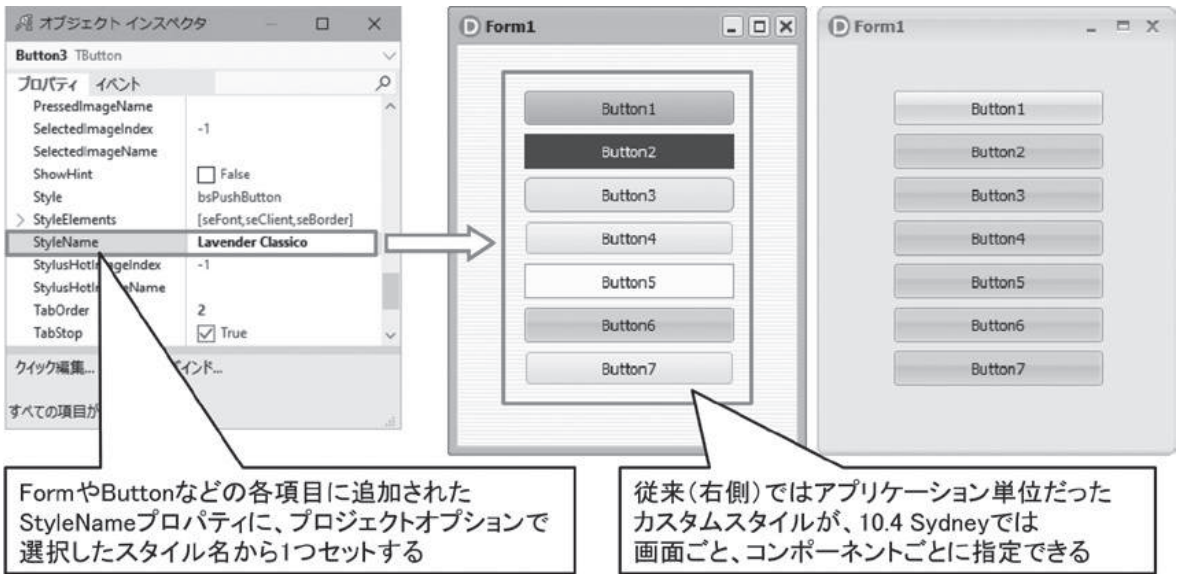


図11　非ビジュアルコンポーネントの表示切替

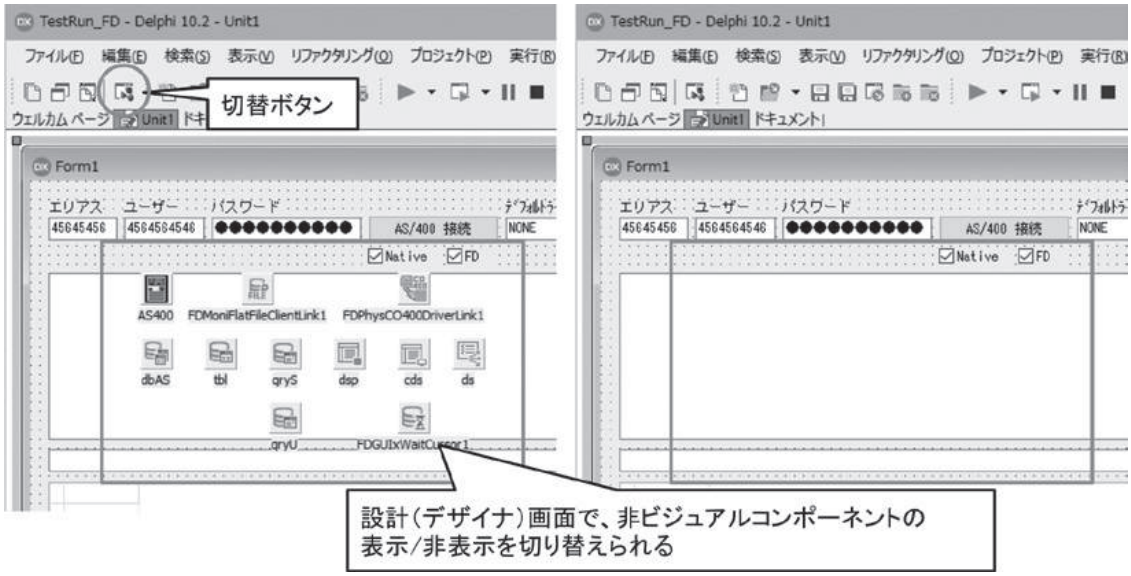


図12　Windows10の見た目を提供するコンポーネント

図13　Windows10の見た目を提供するコンポーネント

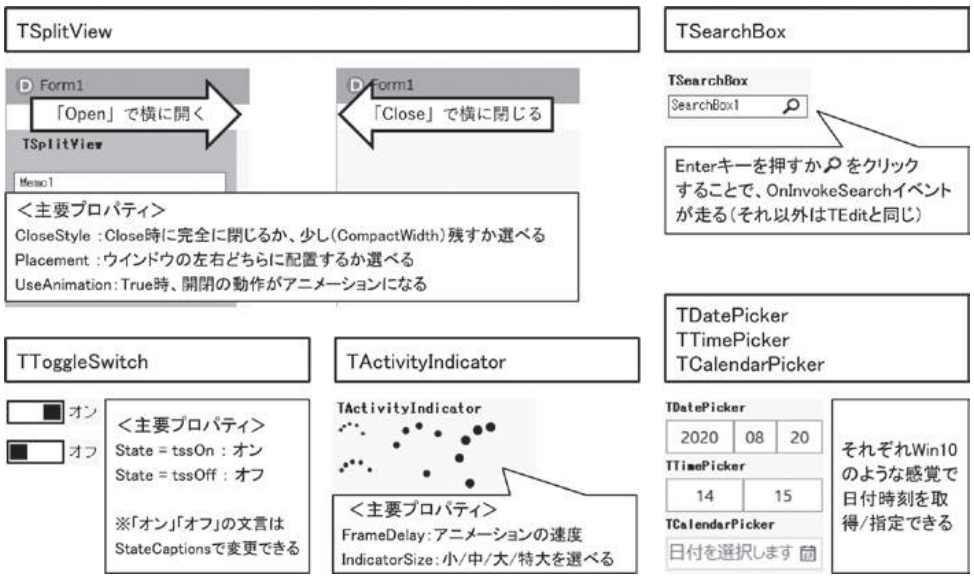


図14　TTitleBarPanelの設定①

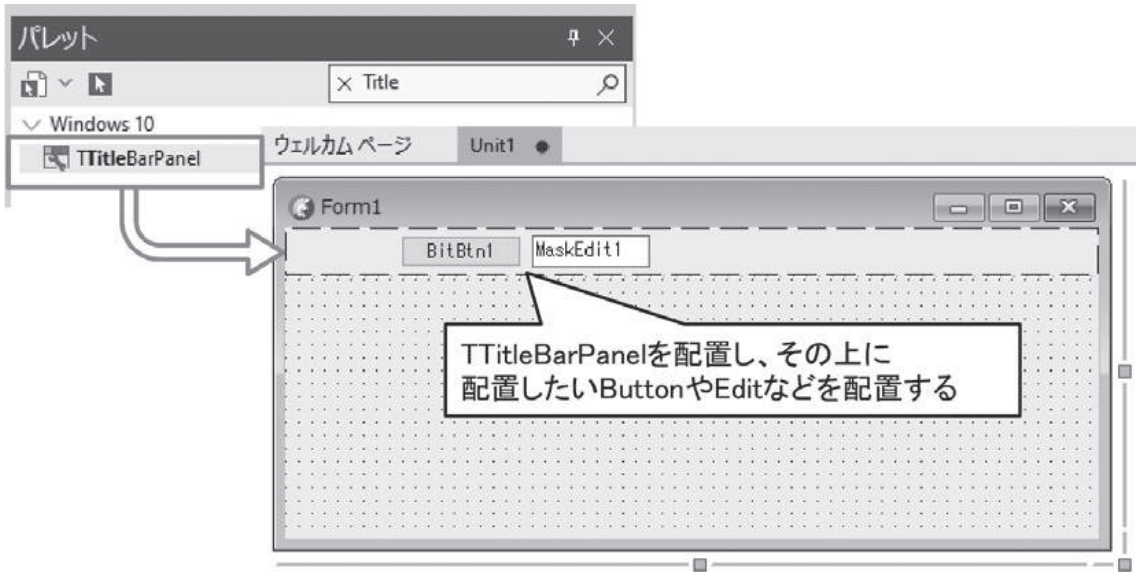


図15　フォームのCustomTitleBarプロパティの指定

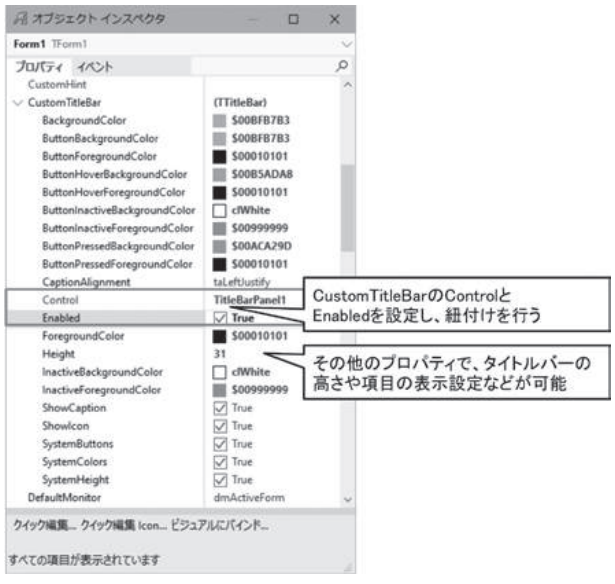


図16　TTitleBarPanelの設定②

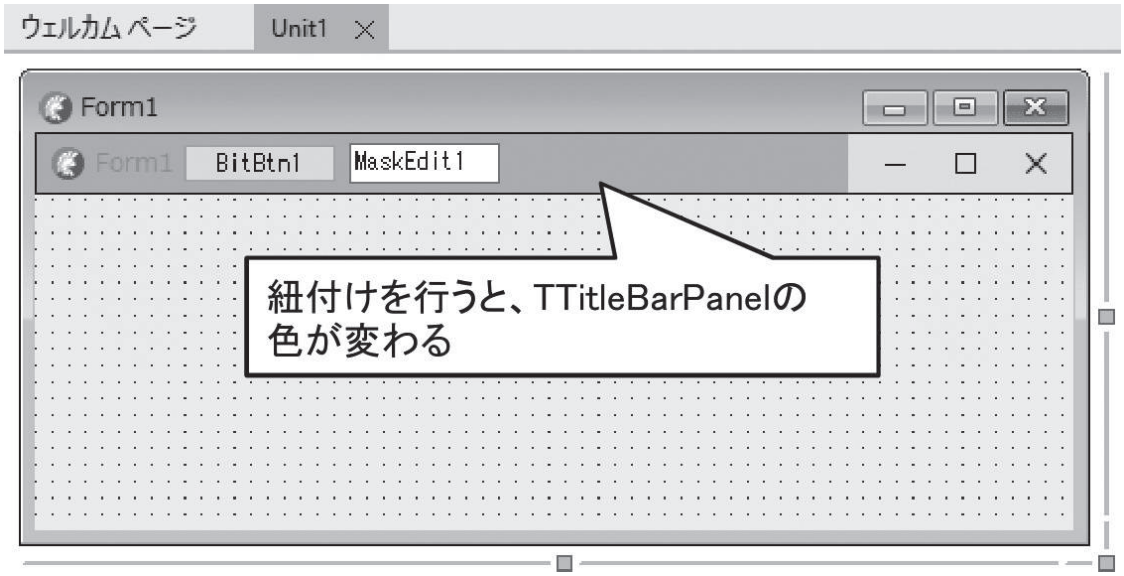


図17　TTitleBarPanelの設定～実行結果

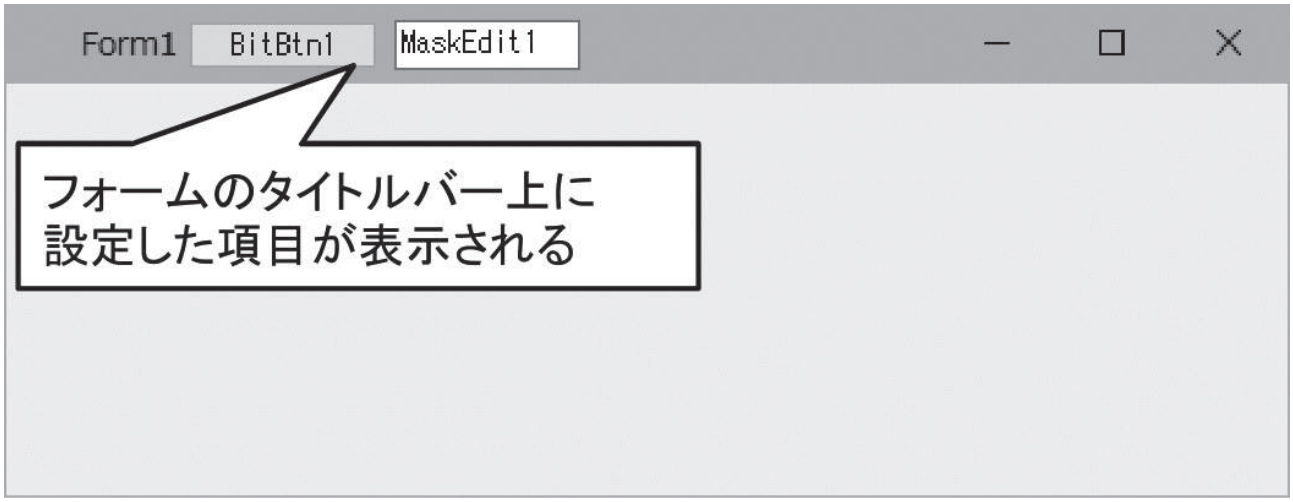
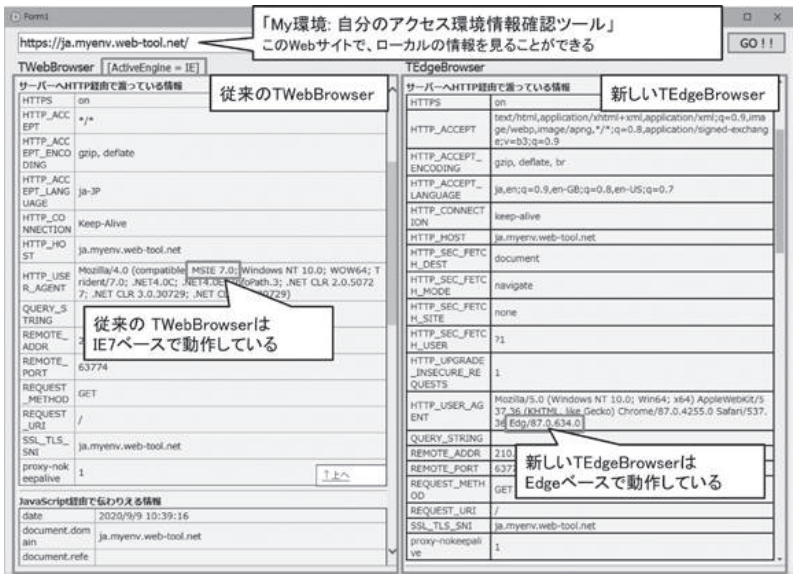


図18　新しいTEdgeBrowser



ソース3

TWebBrowser・TEdgeBrowserでURLを開く

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  // それぞれのコンポーネントでURLを開く
  EdgeBrowser1.Navigate(Edit1.Text);
  WebBrowser1.Navigate(Edit1.Text);
end;
```

どちらも「Navigate」メソッドで
URLにジャンプする

図19　TEdgeBrowserの使用準備①

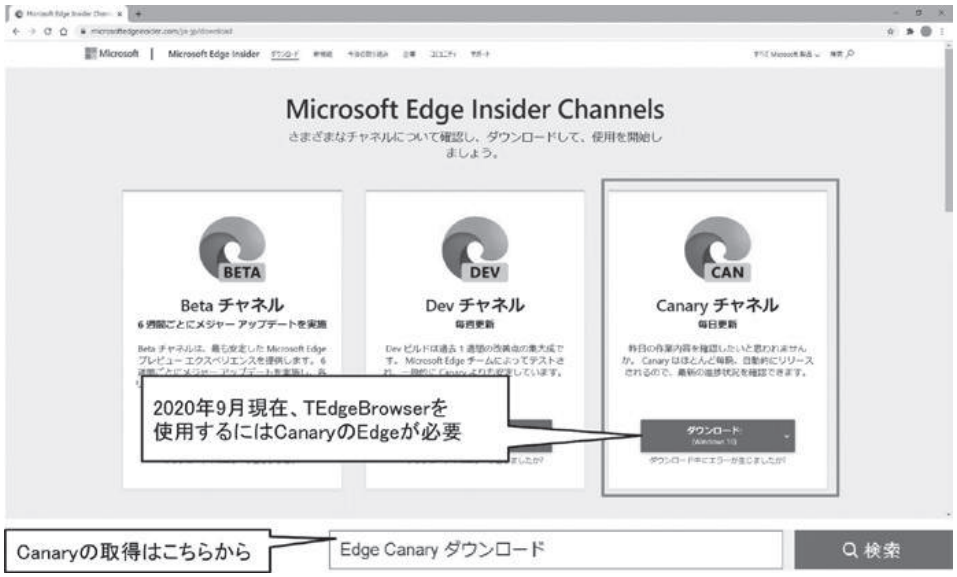


図20　TEdgeBrowserの使用準備②-1

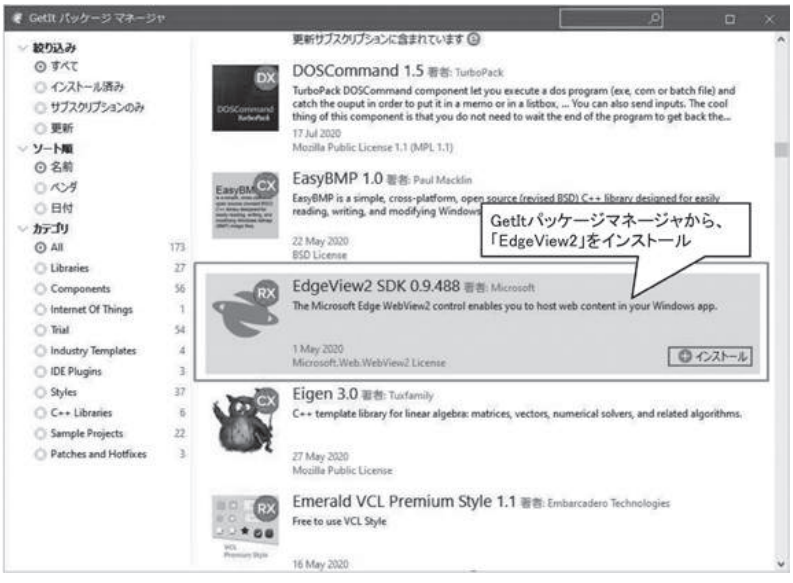


図21　TEdgeBrowserの使用準備②-2

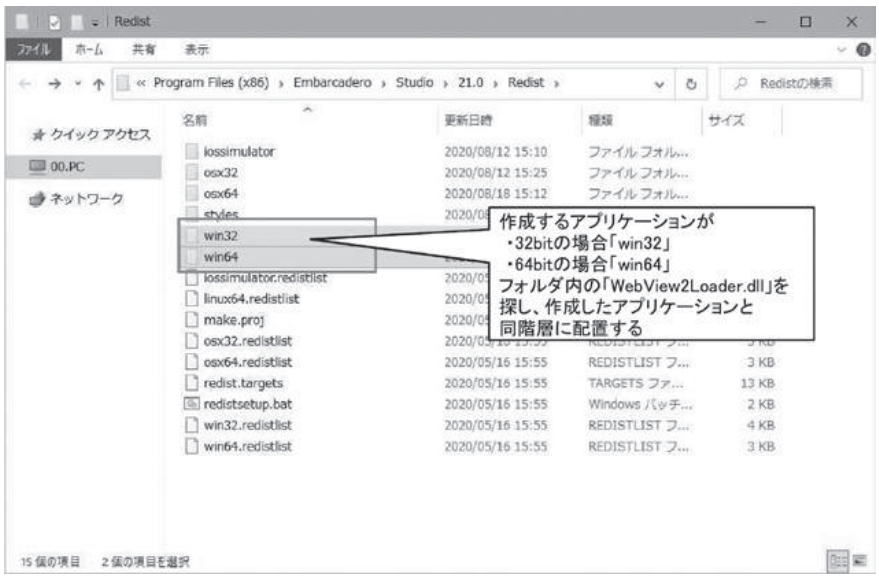


図22　TWebBrowserもEdgeベースで表示可能

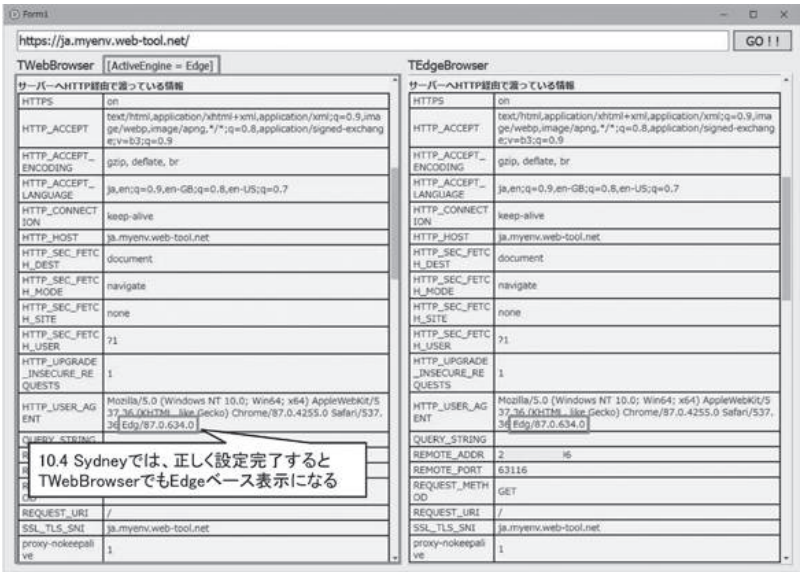


図23　TWebBrowserではEdge互換のない項目がある