LabVIEW™アップグレードノート

このアップグレードノートは、Windows、OS X、およびLinux用のLabVIEWをLabVIEW 2016にアップグレードする方法を説明します。アップグレードを行う前に、以下のトピックの情報について、このドキュメントを参照してください。

- LabVIEWをアップグレードする際の推奨プロセス
- LabVIEWの旧バージョンで保存されたVIをロードする前に考慮するべき互換性の問題
- LabVIEW 2016に新たに追加された機能および動作

目次

LabVIEW 2016へのアップグレード

小さなアプリケーションは、LabVIEWの新しいバージョンをインストールしてからVIをロードすることにより新しいバージョンにアップグレードできますが、アップグレード時の問題をできるだけ効率的に検出して修正するには、より綿密なアップグレードプロセスが推奨されます。

ヒント このプロセスは、重要な操作を制御または監視する、長時間のダウンタイムが許されない、複数のモジュール、ツールキット、ドライバを使用する、またはサポートされないバージョンのLabVIEWで保存された大規模なLabVIEWアプリケーションで特に効果があります。現在主要サポートの対象となっているLabVIEWのバージョ

ンについては、ナショナルインスツルメンツのウェブサイト（ni.com/jp/info）で「lifecycle」と入力してください。

推奨アップグレードプロセスの概要

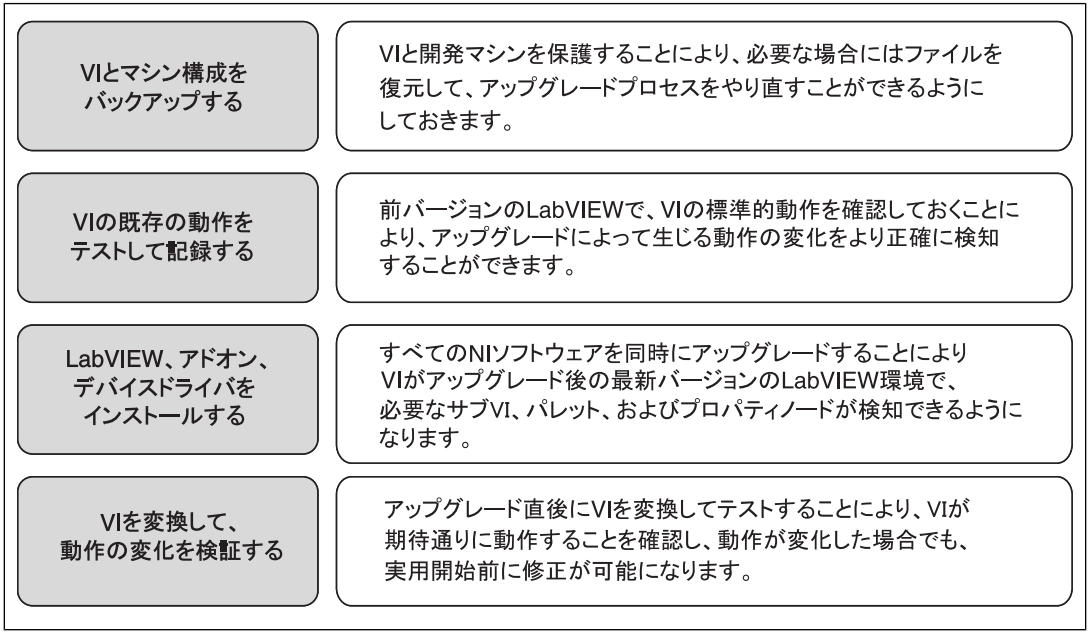

図 1

メモ LabVIEW 5.1以前からアップグレードするには、まずLabVIEWの中間バージョンにアップグレードする必要があります。LabVIEWの特定のレガシーバージョンからのアップグレードについては、ナショナルインスツルメンツのウェブサイト（ni.com/jp/info）で「upgradeOldja」と入力してください。

1. VIおよびマシン構成のバックアップを作成する

必要に応じてVIを以前の機能に復元してアップグレードプロセスを再度実行できるように、LabVIEW 2016にアップグレードする前に、VIのコピーおよび開発マシンまたは本稼動マシンの構成（可能な場合）を保護しておきます。

a. VIのバックアップを作成する

LabVIEWをアップグレードする前にVIをバックアップしておくと、バックアップコピーの状態にすぐ戻すことができます。バックアップコピーがない場合、各VIを旧バージョン用に保存しないと、アップグレードしたVIを旧バージョンのLabVIEWで開けなくなります。

VIセットのバックアップは、以下のいずれかの方法で作成できます。

- **VIをソースコード管理プロバイダにサブミットする**—これにより、VIのアップグレードによる動作の変化に対応できない場合に、VIをこのバージョンに戻すことができます。LabVIEWでソースコード管理プロバイダを使用する方法については、『LabVIEWヘルプ』の**目次**タブで**基本機能→プロジェクトとターゲットを操作する→概念→LabVIEWでソース管理を使用する**トピックを参照してください。

- **VIのコピーを作成する**—VIの構成方法に従って、VIのコピーを作成します。
  - プロジェクトとして保存されている—プロジェクトを開き、**別名で保存**を選択して、`.lvproj`ファイルおよびすべてのプロジェクト内容を複製します。プロジェクトが依存するファイルのコピーも維持するために、**すべての依存項目を含む**を選択してください。
  - LLBまたはディレクトリ内のVIとして保存されている—オペレーティングシステムのファイルエクプローラでLLBまたはディレクトリのコピーを作成し、元の場所とは別の場所に保管します。名前が競合する可能性を回避するために、コピーを同じハードドライブに保管しないようにしてください。

b. マシン構成のバックアップを作成する

新しいバージョンのLabVIEWをインストールすると共有ファイルが更新されるため、旧バージョンのVIの動作も影響を受けることがあります。しかし、それらの共有ファイルを更新後に前のバージョンに戻すことは非常に困難です。このため、サポートされていないバージョンのLabVIEWからアップグレードする場合、またはアプリケーションのダウンタイムコストが高い場合には特に、以下のいずれかの方法で、NIソフトウェアの構成をバックアップすることを検討してください。

- **マシン構成のバックアップイメージを作成する**—アップグレードする前に、インストールされているソフトウェア、ユーザ設定、ファイルなどのマシンのディスク状態を、ディスクイメージングソフトウェアを使用して保存します。アップグレードした後でマシンを元の構成に戻すには、バックアップしたディスクイメージをデプロイします。
- **テストマシンでアップグレードプロセスをテストする**—テストマシンでアップグレードを行うとバックアップイメージを作成するよりも時間はかかりますが、実際のデータ処理を制御または監視するマシンでダウンタイムが発生することを回避または最小化する必要がある場合には、このアプローチを強くお勧めします。アップグレードに起因するすべての問題をテストマシンで解決した後、テストマシンを本稼動マシンに置き換えるか、本稼動マシンでこのアップグレードプロセスを再実行します。

  ヒント アップグレードしたVIがテストマシンと開発マシンで異なる動作をする可能性を最小化するため、開発マシンのCPU、RAM、オペレーティングシステム、ソフトウェアのバージョンなどの装備とできるだけ一致したテストマシンを使用してください。

2. VIの既存の動作をテストして記録する

VIをアップグレードする際、LabVIEWの旧バージョンとLabVIEW 2016の間の違いによってVIの動作が変化する場合があります。両方のバージョンでVIをテストしてその結果を比較することで、アップグレードに起因する動作の変化を検知できます。このため、以下のいずれかのテストの現在の結果が手元にあることを確認してください。

- 一括コンパイルログ—旧バージョンのLabVIEWのVIを一括コンパイルすると、壊れているVIの詳細なログが生成されます。この情報は、複数の人がそのVIの開発に関わっていたり、それらのVIの中に最近コンパイルされていないVIがある可能性がある場合に特に有益です。この一括コンパイルログを生成するには、**一括コンパイル**ダイアログボックスで**結果をログ**チェックボックスをオンにします。VIの一括コンパイルの詳細については、『LabVIEWヘルプ』の**目次**タブで**基本機能→VIおよびサブVIを作成する→操作手順→VIを保存する→VIの一括コンパイル**トピックを参照してください。
- 各VIが意図した機能を正しく実行しているかどうかを検証するユニットテスト
- プロジェクトまたはサブVIのグループがまとまって予期した動作をしているかどうかを検証する統合テスト

- VIがデスクトップやFPGAターゲットなどのターゲットにデプロイされたときに予想どおりに動作するかどうかを検証するデプロイメントテスト
- CPU使用率、メモリ使用、コード実行速度のベンチマーク測定する性能テスト。**パフォーマンスおよびメモリをプロファイル**ウィンドウを使用すると、VIの平均実行速度を計算できます。
- VIが予期しないデータを正しく処理できるかどうかを検証するストレステスト

VIテストの詳細については、『LabVIEWヘルプ』の**目次**タブの**基本機能→アプリケーション開発と設計ガイドライン→概念→大規模アプリケーションを開発する→開発モデルの段階→アプリケーションをテストする**トピックを参照してください。

メモ　テスト結果をもとにVIを変更した場合は、この先に進む前に新しいバージョンのVIをバックアップします。

3. LabVIEW、アドオン、およびデバイスドライバをインストールする

a. LabVIEW（モジュール、ツールキット、およびドライバを含む）をインストールする

新しいバージョンのLabVIEWにアップグレードした場合、新しい開発システムだけでなく、その新しいバージョンと互換性のあるモジュール、ツールキット、およびドライバをインストールする必要があります。このソフトウェアを正しい順序でインストールする方法については、『LabVIEWインストールガイド』を参照してください。

b. user.libファイルをコピーする

旧バージョンのLabVIEWで作成したカスタム制御器とVIをLabVIEW 2016で使用するには、旧バージョンのLabVIEWの`labview¥user.lib`ディレクトリのファイルをLabVIEW 2016の`labview¥user.lib`ディレクトリにコピーします。

4. VIを変換し、動作の変化を検証する

LabVIEW 2016でVIを一括コンパイルすると、VIはLabVIEWの最新バージョンに変換され、壊れたVIの特定に役立つエラーログが作成されます。この情報を、このドキュメントの「アップグレードおよび互換性の問題」セクションと併せて使用することにより、LabVIEWの最新バージョンに関連する動作変化を特定して修正できます。

a. VIを新しいバージョンのLabVIEWで一括コンパイルする

VIを一括コンパイルすると、VIはLabVIEW 2016に変換されて保存されます。しかし、各VIまたはプロジェクトに対して**ファイル→旧バージョン用に保存**を選択しない限り、VIを旧バージョンのLabVIEWで開けなくなります。このため、新しいバージョンのLabVIEWに変換したいVIのみを一括コンパイルしてください。アップグレードにより発生した問題を特定するには、**一括コンパイル**ダイアログボックスの**結果をログ**チェックボックスをオンにして一括コンパイルログを作成します。

メモ　FPGAまたはリアルタイムリソースが含まれているVIを一括コンパイルすると、**一括コンパイル**ダイアログボックスに、これらのVIは実行不可能なVIであるとレポートされる場合があります。エラーを確認するには、FPGAまたはRTターゲットの下にあるVIを、プロジェクト内で必要なFPGAリソースまたはリアルタイムリソースとともに開きます。

VIの一括コンパイルの詳細については、『LabVIEWヘルプ』の**目次**タブで以下のトピックを参照してください。

- **基本機能→VIおよびサブVIを作成する→操作手順→VIを保存する→VIを一括コンパイルする**

- **基本機能→VIおよびサブVIを作成する→操作手順→VIを保存する→一括コンパイルの一般的なステータスメッセージ**

b. 壊れたVIを修正する

旧バージョンのLabVIEWとLabVIEW 2016との違いが原因で、古い機能を使用している一部のVIが壊れる場合があります。壊れたVIをLabVIEW 2016で特定して、修正するには、以下の手順を実行してください。

1. アップグレード中に壊れたVIを特定するには、前のステップで作成した一括コンパイルエラーログと、VIの既存の動作を確認するために作成したログを比較します。
2. 各VIがLabVIEWのどのアップデートが原因で壊れたかを調べるには、このドキュメントの「アップグレードおよび互換性の問題」セクションを参照してください。

c. 動作の変化を特定して修正する

ナショナルインスツルメンツでは、LabVIEWのバージョン間におけるVIの動作の違いを最小限に抑えるために多大な努力を払っておりますが、VIの改善やバグ修正の結果、VIの動作が変化することがあります。LabVIEWの新しいバージョンでVIの動作が変更するかどうかを素早く確認するには、以下のツールを使用します。

- **アップグレードVIアナライザテスト**—大規模なVIセットの場合、これらのテストにより、アップグレードに起因する多くの動作変化を効率的に特定できます。これらのテストを取得して使用するには、以下の手順に従います。
  1. 旧バージョンのLabVIEW以降のすべてのバージョンのLabVIEWのアップグレードVIアナライザテストをダウンロードします。これらのテストをダウンロードするには、ナショナルインスツルメンツのウェブサイト（`ni.com/jp/info`）でInfo Codeに「`analyzevi`」を入力してください。
  2. **ツール→VIアナライザ→VIを解析**を選択して、新規のVIアナライザタスクを開始することにより、テストを開いて、実行します。プロジェクト全体をまとめて解析するには、個々のVIからではなく、**プロジェクトエクスプローラ**ウィンドウでこのメニュー項目を選択します。
  3. テストに失敗した場合は、テストに対応するLabVIEWのバージョンの「アップグレードおよび互換性の問題」セクションを参照して修正します。たとえば、LabVIEW 2013アップグレードVIアナライザテストで、変化する可能性のある動作が見つかった場合は、そのトピックの「LabVIEW 2012からアップグレードする」セクションを参照します。
- **アップグレードドキュメント**
  - このドキュメントのアップグレードおよび互換性の問題トピック—VIを壊したり、VIの動作に影響を与える可能性のある変更が記載されています。前のバージョンから始めて、LabVIEWの各バージョンのサブセクションを参照します。

    ヒント 「アップグレードおよび互換性の問題」セクションで言及されている廃止されたオブジェクトやその他のオブジェクトを手早く見つけるには、アップグレードしたVIを開き、**編集→検索して置換**を選択します。

  - LabVIEW 2016既知の問題リスト—LabVIEW 2016のリリース前およびリリース中に発見されたバグが記載されています。このリストを参照するには、ナショナルインスツルメンツのウェブサイト（`ni.com/jp/info`）でInfo Codeに「`lv2016ki`」を入力してください。「アップグレード—動作の変化」と「アップグレード—移行」セクションがある場合は、アップグレードしたVIの動作に影響するバグの回避方法についてそれらを参照してください。

- モジュールおよびツールキットのドキュメント—LabVIEW FPGAおよびLabVIEW Real-Timeモジュールなど、一部のモジュールおよびツールキットに特有のアップグレード問題が記載されています。
- ドライバのReadmeファイル—各ドライバに特有のアップグレード問題が記載されています。各Readmeを見つけるには、そのドライバのインストールメディアを参照してください。

ヒント 動作が変化した原因がLabVIEWのアップデートではなく、ドライバのアップデートであることを確認するには、LabVIEW 2016をインストールした後に旧バージョンのLabVIEWでそのVIをテストしてください。

- **ユーザ独自のテスト**—旧バージョンのLabVIEWで行ったのと同じテストをLabVIEW 2016のVIに対して行い、結果を比較します。新しい動作を特定した場合は、その変化の原因を診断するために、アップグレードドキュメントを参照します。

アップグレードに関する一般的な問題のトラブルシューティング

アップグレードに関する以下の問題の解決方法については、『LabVIEWヘルプ』の**目次**の**LabVIEW 2016へのアップグレード→アップグレードに関する一般的な問題のトラブルシューティング**トピックを参照してください。

- 見つからないモジュールまたはツールキットの機能を探す
- 見つからないサブVI、パレット、プロパティノードを探す
- 旧バージョンのLabVIEWで作成したVIがLabVIEW 2016で開けない理由を特定する
- インストールされているNIソフトウェアのバージョンを特定する
- VIを旧バージョンのLabVIEWに復元する

アップグレードおよび互換性の問題

VIを壊したり、動作を変更したりする可能性のある、LabVIEWのバージョンごとの変更については、以下のセクションを参照してください。

新しいバージョンのLabVIEWの既知の問題、その他の互換性に関する問題、LabVIEW 2016の最新の追加機能の詳細は、`labview`ディレクトリの`readme.html`を参照してください。

LabVIEW 2011以前からアップグレードする

LabVIEW 2011以前からLabVIEW 2016へのアップグレード時に発生する可能性があるアップグレードまたは互換性の問題に関する情報を参照するには、ナショナルインスツルメンツのウェブサイト（`ni.com/jp/info`）で「`upnote12jp`」と入力してください。また、その他のアップグレードに関する問題については、このドキュメントの「LabVIEW *x*からアップグレードする」セクションを参照してください。

LabVIEW 2012からアップグレードする

LabVIEW 2012からLabVIEW 2016へアップグレードすると、以下の互換性問題が発生する可能性があります。発生する可能性のあるその他のアップグレード関連の問題については、このドキュメントの「LabVIEW 2013からアップグレードする」、「LabVIEW 2014からアップグレードする」、および「LabVIEW 2015からアップグレードする」セクションを参照してください。

VIと関数の動作の変更点

以下のVIおよび関数の動作はLabVIEW 2013で変更されました。

ウェブサービスVI

ウェブサービスパレットの以下のVIは、LabVIEW 2013で改訂されました。VIの**httpRequestID**入力は**LabVIEWウェブサービス要求**入力に変更されました。新しい機能を使用するには、廃止されたVIをウェブサービスパレットにある同名のVIに置き換えてください。

- ウェブサービスパレット:
  - すべてのフォームデータを読み取る
  - すべての要求変数を読み取る
  - フォームデータを読み取る
  - ポストデータを読み取る
  - 要求変数を読み取る
  - アップロードされたファイルの情報を読み取る
- 出力サブパレット:
  - 出力を排出
  - ESPテンプレートをレンダリング
  - ESP変数を設定
  - HTTPヘッダを設定
  - HTTPリダイレクトを設定
  - HTTP応答コードを設定
  - HTTP応答MIMEタイプを設定
  - 応答を書き込む
- セキュリティサブパレット:
  - 復号化
  - 暗号化
  - 詳細な認証情報を取得
- セッションサブパレット:
  - セッションが存在するかどうかをチェック
  - セッションを作成
  - セッション変数を削除
  - セッションを破棄
  - セッションIDクッキーを取得
  - すべてのセッション変数を読み取る
  - セッション変数を読み取る
  - セッション変数を書き込む

ダイナミック登録されている未処理のイベントに対するイベントストラクチャタイムアウト端子の動作に関する変更

LabVIEW 2012以前では、イベントをダイナミック登録した場合、イベントストラクチャで処理が構成されていないイベントが発生したときに、タイムアウト端子がそのイベントによってリセットされる場合があります。たとえば、「イベント登録」関数を使用してマウスアップ、マウスダウン、マウス移動の各イベントが登録されている場合に、イベントストラク

チャでマウスアップとマウスダウンのイベントのみを登録すると、マウス移動イベントが発生した場合にタイムアウト端子がリセットされます。

メモ タイムアウト端子がリセットされるのは、その端子に値を配線した場合のみです。

LabVIEW 2013では、ダイナミック登録されたイベントの処理が定義されていなくても、そのイベントが発生したときにイベントストラクチャのタイムアウト端子はリセットされません。

デフォルト.NET Frameworkに関する変更

LabVIEW 2013で.NETオブジェクトを作成して通信するには、.NET Framework 4.0が必要です。.NET Framework 4.0では、.NET Frameworkの任意のバージョンでビルドされた純粋なマネージアセンブリ、および.NET 4.0でビルドされたミックスモードアセンブリをロードできます。LabVIEW 2013インストーラには、.NET Framework 4.0が含まれています。しかし、もし.NET Framework 4.0をアンインストールしたり、別のバージョンの.NET Frameworkをターゲットにしたアセンブリをロードしようとすると、.NETオブジェクトを作成して通信しようとしたときにエラーが返されることがあります。

LabVIEW 2013は、デフォルトで共通言語ランタイム（CLR）4.0をロードします。しかし、CLR 2.0をターゲットにした.NETミックスモードアセンブリを強制的にロードさせることができます。

LabVIEWでのアセンブリのロードについては、『LabVIEWヘルプ』の**目次**タブで**基本機能→Windowsコネクティビティ→操作手順→.NET→LabVIEWで.NET 2.0、3.0、3.5アセンブリをロードする**を参照してください。

システムボタンに関する変更

LabVIEW 2012以前では、**システム**パレットからフロントパネルにシステムボタンを追加した場合、デフォルトでは、改行キーで値が切り替わります。LabVIEW 2013では、システムボタンにデフォルトキーバインディングは適用されません。

値および値（信号）プロパティに関する変更

LabVIEW 2012以前では、値プロパティまたは値（信号）プロパティを使用してラッチ型ブール制御器の値を設定すると、LabVIEWはエラーを返します。しかし、ラッチ型ブール制御器をタイプ定義に変更すると、エラーが返されなくなります。LabVIEW 2013では、ラッチ型ブール制御器の値を設定しようとすると、値プロパティまたは値（信号）プロパティは、競合状態を回避するために、常にエラーを返します。

条件トンネルのパフォーマンスの改善点

LabVIEW 2012では、ループの各出力トンネルで指定する値のみを含ませるために**条件**トンネルオプションを使用することができますが、ナショナルインスツルメンツではパフォーマンスが重要なアプリケーションにおいては条件トンネルに代わる別のオプションを使用することを推奨します。LabVIEW 2013では、条件トンネルのパフォーマンスを向上させるために、**条件**トンネルオプションへのメモリ割り当てが削減されます。

カスタム制御器をサブパネルに配線する

サブパネルクラスのVIを挿入メソッドにカスタム制御器を配線すると、エラーが返されます。カスタム制御器をサブパネルに配線するには、VIのフロントパネルに制御器を追加して、そのVIをサブパネルに配線します。

NI Web-Based Configuration & MonitoringをSSL付きで使用する

LabVIEW 2012以前では、SSL（Secure Sockets Layer）証明書およびCSR（証明書署名要求）をNI分散システムマネージャから表示して編集します。この機能は、分散システムマネージャでサポートされなくなりました。

SSL証明書とCSRの作成、編集、表示、削除は、NI Web-based Configuration & Monitoringから行えるようになりました。SSL証明書とCSRを管理するには、NI Web-based Configuration & Monitoringユーティリティでウェブサーバ構成ページに移動し、SSL証明書管理タブを表示します。

LabVIEWウェブサービスの作成とパブリッシュ

LabVIEW 2013では、ウェブサービスを作成したり、ウェブサービスのプロパティ（URLマッピングなど）を構成する際、RESTfulウェブサービスビルド仕様を使用しなくなりました。LabVIEW 2012以前で作成したビルド仕様は、引き続き使用するか、ウェブサービスプロジェクト項目に変換できます。変換ツールをダウンロードするには、`ni.com/jp/info`で「`ConvertWS`」と入力してください。

ウェブサービスをLabVIEW 2013形式に変換する場合、プロジェクトのウェブサービスプロジェクト項目を右クリックして、**プロパティ**を選択することにより、ウェブサービスビルド仕様を構成するためにLabVIEW 2012以前で提供されていたほとんどのオプションにアクセスできます。LabVIEW 2012以前で使用できたウェブサービス動作とオプションのうち、LabVIEW 2013で変更または削除されたものを以下に表で示します。

LabVIEW 2012以前	LabVIEW 2013
ウェブメソッドVIとは、クライアントからHTTP要求を受け取り、クライアントにデータを返すVIのことを意味します。	ウェブメソッドVIの概念は、「HTTPメソッドVI」という名前に変更されました。
ウェブサービス名のサービスエイリアスを定義することにより、クライアントがサービスにアクセスする際のURLをカスタマイズすることが可能です。	ウェブサービスにアクセスするには、同一のサービス名を使用します。
ウェブメソッドVIに複数のURLのマッピングが可能です。	HTTPメソッドVIにマッピングできるのは1つのURLのみです。複数のURLが同じVIを呼び出すことを許可するには、複数のHTTPメソッドVI内でサブVIとして使用します。各URLマッピングは各HTTPメソッドVIに保持します。
VIのコネクタペーン端子のデフォルト値をオーバーライドする値を指定することが可能です。	複数のURLを1つのHTTPメソッドVIにマッピングすることは許可されないため、このオプションは削除されました。このため、オーバーライド動作に依存する代替URLマッピングを作成することはできません。
プロジェクト内のVIは代替VIとしてマークすることができます。代替VIは、ウェブメソッドVIとデータを交換するが、クライアントには公開されません。	代替VIの概念は、スタートアップVIという名前に変更されました。LabVIEWは、プロジェクトで**スタートアップVI**プロジェクト項目の下に配置されたすべてのVIをスタートアップVIとみなします。
ウェブサービスの「スタンドアロン」デプロイメントを無効にすることが可能です。この場合、ウェブサービスはLabVIEW開発システムがオープンのときにのみデプロイされます。	このオプションは削除されました。
VIを、ウェブサービスのビルド時に実行されるビルド前ステップまたはビルド後ステップとして実行されるように設定することが可能です。	ウェブサービスはビルド仕様からビルドしないため、このオプションは使用できません。

キューメッセージハンドラテンプレートへの変更

キューメッセージハンドラテンプレートのエラー処理スキームは、LabVIEW 2013で変更されました。新しいエラー処理スキームでは、各ループがループ専用のエラー処理サブVIを使っ

てエラーを処理します。メッセージ処理ループでエラーが発生した場合、LabVIEWはエラーメッセージを表示します。

連続測定とログサンプルプロジェクトへの変更

連続測定とログサンプルプロジェクトのエラー処理スキームは、LabVIEW 2013で変更されました。新しいエラー処理スキームでは、各ループがループ専用のエラー処理サブVIを使ってエラーを処理します。メッセージ処理ループでエラーが発生した場合、LabVIEWはエラーメッセージを表示します。

LabVIEW 2013以降では、集録メッセージループとログメッセージループに、ループがすでに開始している場合は開始メッセージ、すでに停止している場合は停止メッセージを受信するケースを処理するためのステートが含まれています。

LabVIEW 2013からアップグレードする

LabVIEW 2013からLabVIEW 2016へアップグレードすると、以下の互換性問題が発生する可能性があります。発生する可能性のあるその他のアップグレード関連の問題については、このドキュメントの「LabVIEW 2014からアップグレードする」および「LabVIEW 2015からアップグレードする」セクションを参照してください。

「文字列をパスに変換」関数の動作変更

LabVIEW 2014以降では、「文字列をパスに変換」関数は、<Not A Path>文字列の何らかのバリエーションを読み取った時に大/小文字を区別せず、常に<無効パス>定数を返します。たとえば、**文字列**入力には<not a path>または<Not A Path>と指定でき、いずれの場合も、この関数は<無効パス>という値の定数を返します。旧バージョンのLabVIEWでの「文字列をパスに変換」関数の動作については、下表を参照してください。

LabVIEW 2012および2013	LabVIEW 2011以前
「文字列をパスに変換」関数は、大/小文字に関係なく、<無効パス>定数を返しません。<Not A Path>のあらゆるバリエーションが指定でき、この関数は、<無効パス>定数を返すのではなく、<Not A Path>という名前のディレクトリのパスを返します。	LabVIEW 2014以降と同様、「文字列をパスに変換」関数は、大/小文字を区別せず、<Not A Path>文字列の何らかのバリエーションが指定されたときには、<無効パス>定数を出力します。<not a path>と指定された場合も、<Not a Path>と指定された場合も、この関数は、<無効パス>定数を返します。

タイプ定義を確認して更新する

LabVIEW 2013以前で表示される**タイプ定義から更新**ショートカットメニュー項目は、**確認してタイプ定義から更新**ショートカットメニュー 項目に変更されました。

廃止されたVI、関数、およびノード

LabVIEW 2014以降は、以下のVI、関数、ノードをサポートしていません。

AppleイベントVI

（OS X）LabVIEW 2014以降は、AppleイベントVIをサポートしていません。LabVIEW外部のOS Xアプリケーションと通信するには、ライブラリ&実行可能ファイルパレットの「Run AppleScript Code」VIを使用してください。以下のいずれかのAppleイベントVIを含むVIをロードしようとすると、LabVIEWでエラーが発生し、VIが実行できない可能性があります。

- AESend Do Script
- AESend Finder Open
- AESend Open
- AESend Open Document
- AESend Print Document

- AESend Quit Application
- Get Target ID
- AESend Abort
- AESend Close
- AESend Open、Run、Close
- AESend Run
- AESend VI Active?
- AECreate Comp Descriptor
- AECreate Descriptor List
- AECreate Logical Descriptor
- AECreate Object Specifier
- AECreate Range Descriptor
- AECreate Record
- AESend
- Make Alias

アクターフレームワークVI

LabVIEW 2014以降は、「アクター:アクターを起動」VIをサポートしていません。代わりに、「アクター:ルートアクターを起動」VIまたは「アクター:ネストされたアクターを起動」VIを使用してください。

「In Port」VIおよび「Out Port」VI

LabVIEW 2014以降は、「In Port」VIおよび「Out Port」VIをサポートしていません。

廃止されたプロパティ、メソッド、およびイベント

LabVIEW 2014以降は、アプリケーションクラスの古いヘルプ情報メソッドをサポートしていません。指定されたVIの**VIプロパティ**ダイアログボックスの**ドキュメント**ページからヘルプ情報を返すには、代わりにVIを取得:ヘルプ情報メソッドを使用してください。

LabVIEW 2014からアップグレードする

LabVIEW 2014からLabVIEW 2016へアップグレードすると、以下の互換性問題が発生する可能性があります。発生する可能性のあるその他のアップグレード関連の問題については、このドキュメントの「LabVIEW 2015からアップグレードする」セクションを参照してください。

LabVIEWアプリケーションにおけるバッファ割り当てを識別する

LabVIEW 2014 Service Pack 1以降には、LabVIEWアプリケーションにおけるバッファ割り当てを特定および解析するための**バッファ割り当てをプロファイル**ウィンドウが追加されました。このウィンドウを表示するには、**ツール→プロファイル→バッファ割り当てをプロファイル**を選択します。

フリーラベル内のハイパーリンク

LabVIEW 2015以降では、LabVIEWがフリーラベル内のURLを検出し、下線付きの青いテキストのハイパーリンクに変換します。LabVIEW 2014以前からアップグレードした場合、フリーラベル内のURLは自動的にはハイパーリンクに変換されません。フロントパネルのラベルのハイパーリンクを有効にするには、フリーラベルを右クリックし、ショートカットメニューで**ハイパーリンクを有効化**を選択します。ブロックダイアグラムのラベルでハイパーリンクを無効にすることはできません。

廃止されたVI、関数、およびノード

LabVIEW 2015以降は、以下のVI、関数、およびノードをサポートしていません。

- 「**スプレッドシートファイルから読み取る**」—代わりに「区切られたスプレッドシートを読み取る」VIを使用します。
- 「**スプレッドシートファイルに書き込む**」—代わりに「区切られたスプレッドシートに書き込む」VIを使用します。

LabVIEW 2015からアップグレードする

LabVIEW 2015からLabVIEW 2016へアップグレードすると、以下の互換性問題が発生する可能性があります。

LabVIEW 2016以降では、**クイックドロップ構成**ダイアログボックスに、フロントパネルオブジェクトとブロックダイアグラムオブジェクトのショートカットのデフォルトリストが含まれています。LabVIEW 2015以前で作成したショートカットは、LabVIEW 2016以降のショートカットリストに自動的に統合されません。

LabVIEW 2016の機能および変更点

既知の問題、修正されたバグの部分的なリスト、その他の互換性に関する問題、LabVIEW 2016における追加機能については、`labview`ディレクトリの`readme.html`ファイルを参照してください。

オブジェクトの選択、移動、およびサイズ変更における改善点

LabVIEW 2016では、フロントパネルまたはブロックダイアグラムでのオブジェクトの選択と移動、およびストラクチャのサイズ変更がより簡単に行えるようになりました。

- **オブジェクトを選択する**
  - オブジェクトを選択する際、選択四角形で囲まれた部分はグレー表示され、選択されたオブジェクトはマーキーによりハイライト表示されます。ストラクチャを選択すると、その背景が濃くなり、選択されていることを示します。
  - デフォルトでは、オブジェクトを選択四角形で囲んで選択する場合、ストラクチャ全体またはワイヤセグメントの中間点を囲む必要があります。スペースバーを押しながら選択四角形を作ると、選択四角形が接触したすべてのオブジェクトが選択されます。デフォルト動作に戻すには、スペースバーをもう一度押します。
- **オブジェクトを移動する**—選択されたオブジェクトを移動すると、選択された部分全体がリアルタイムで移動します。ストラクチャなどの一部のオブジェクトは、選択されたオブジェクトの移動に合わせて配置やサイズを変更します。
- **ストラクチャをサイズ変更する**—サイズ変更ハンドルをドラッグしてストラクチャのサイズを変更すると、ストラクチャは破線で表示されずに、リアルタイムで拡大または収縮します。

並列したコードセクション間でデータを非同期通信する

LabVIEW 2016では、チャンネルワイヤを使用して並列したコードセクションの間でデータを通信できます。チャンネルワイヤは、並列した2つのコードセクションを実行順序を強制せずに接続する非同期ワイヤであり、コードセクション間でのデータ依存性を発生させないようにします。

LabVIEWには、それぞれ異なる通信プロトコルを扱う、いくつかのチャンネルテンプレートが含まれています。アプリケーションの通信要件に応じてテンプレートを選択できます。

チャンネルワイヤを作成するには、まず最初に端子またはワイヤを右クリックして**作成**→**チャンネル書き込み**を選択し、書き込みエンドポイントを作成します。書き込みエンドポイ

ントのチャンネル端子からチャンネルワイヤを描き、そのチャンネルワイヤを右クリックして**作成→チャンネル読み取り**を選択し、読み取りエンドポイントを作成します。

書き込みエンドポイントはチャンネルにデータを書き込み、読み取りエンドポイントはチャンネルからデータを読み取ります。Refnumや変数もチャンネルワイヤと同じ方法でコードセクション間のデータ通信を行いますが、チャンネルワイヤを使用した方が、必要なノード数が少なく、目に見えるワイヤを使用してデータ通信を視覚的に表すことができます。

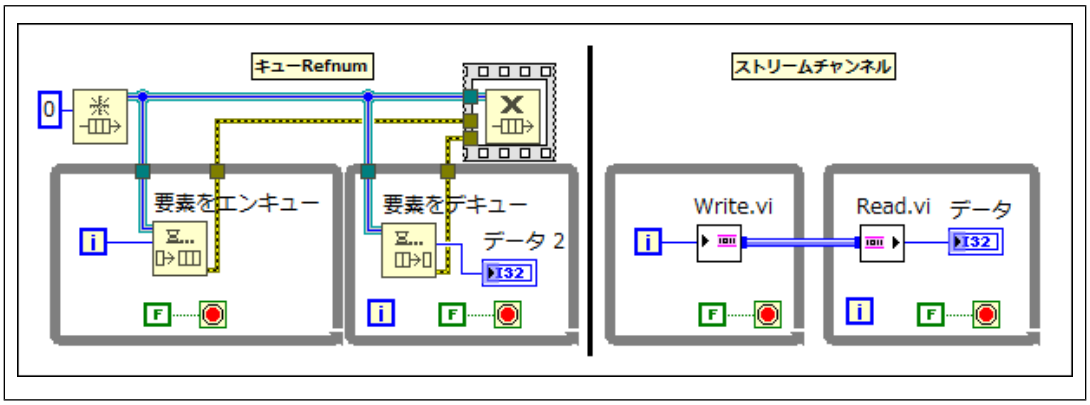

図 2

環境の改善点

LabVIEW 2016では、LabVIEW環境が以下のように改善されました。

ダイアログボックスの改善点

クイックドロップ構成ダイアログボックスに、フロントパネルオブジェクトとブロックダイアグラムオブジェクトのショートカットのデフォルトリストが追加されました。ショートカットを手動で構成する代わりに、**クイックドロップ**ダイアログボックスのこのデフォルトショートカットを使用できます。

クイックドロップ構成ダイアログボックスでは、**フロントパネル**ページおよび**ブロックダイアグラム**ページに以下のオプションも追加されました。

- **デフォルトのパネルショートカットを復元する/デフォルトのダイアグラムショートカットを復元する**—既存のショートカットリストをショートカットのデフォルトリストで置き換えます。
- **すべてのパネルショートカットを削除する/すべてのダイアグラムショートカットを削除する**—リストからすべてのショートカットを削除します。

メモ　デフォルトのパネルショートカットを復元する/デフォルトのダイアグラムショートカットを復元するまたは**すべてのパネルショートカットを削除する/すべてのダイアグラムショートカットを削除する**をクリックした後、**OK**をクリックして変更を適用します。変更を取り消すには、**キャンセル**をクリックします。

（Linux） フォントおよびエンコーディングサポートの改善

Linux用LabVIEW 2016には、True Typeフォントおよび UTF-8文字エンコーディングのサポートが追加されました。

- LabVIEWはデフォルトでTrue Typeフォントを使用します。LabVIEWビットマップフォントに必要であった特別なフォントパッケージをインストールする必要がなくなりまし

た。ビットマップフォントに戻すには、UseXftFonts=False構成トークンを/home/<username>/natinst/.config/<labview>/labview.confにあるLabVIEW構成ファイルに追加します。

- LabVIEWが実行されているLinuxシステムのロケール環境がUTF-8エンコーディングを使用するように構成されている場合、LabVIEWはUTF-8エンコーディングを使用します。LabVIEWでUTF-8エンコーディングを無効にするには、DisableUTF8=True構成トークンを/home/<username>/natinst/.config/<labview>/labview.confにあるLabVIEW構成ファイルに追加します。

追加および変更されたVIおよび関数

LabVIEW 2016には、以下のVIおよび関数が新たに追加されました。

新しいVIおよび関数

LabVIEW 2016には、以下のVIおよび関数が新たに追加されました。

- 上級ファイルパレットに以下のVIが新たに追加されました。
  - **名前がマルチプラットフォームか**—このVIは、ファイル名がLabVIEWでサポートされているオペレーティングシステムで有効かどうかを確認するために使用します。
  - **ファイルシステムで表示**—このVIは、ファイルまたはディレクトリのパスを、現在のプラットフォームに基づいて**（Windows）**Windowsエクスプローラー、**（OS X）**Finder、**（Linux）**ファイルシステムブラウザで開くために使用します。
- データタイプ解析パレットに「チャンネル情報を取得」VIが新たに追加されました。このVIは、チャンネル情報および転送データタイプを取得するために使用します。
- アクターフレームワークパレットは、新しい「自動停止のネストされたアクターカウントを読み取る」VIが新たに追加されました。このVIは、発呼者アクターに最終Ackメッセージを送信していないネストされたアクターの数を返すために使用します。このVIは、発呼者アクターが停止した時に停止するネストされたアクターのみを数えます。
- In Place要素ストラクチャにバリアント属性取得/置換境界ノードが新たに追加されました。この境界ノードを使用すると、バリアントの属性をコピーせずに、バリアントの1つまたは複数の属性にアクセスし、変更できます。バリアント属性取得/置換境界ノードを使用して高性能のルックアップテーブルを作成するサンプルについては、NIサンプルファインダの「バリアント属性ルックアップテーブル」VIまたはlabview¥examples¥Performance¥Variant Attribute Lookup Tableディレクトリを参照してください。

変更されたVIおよび関数

LabVIEW 2016では、以下のVIおよび関数が変更されました。

- **（Windows）** OS X用およびLinux用のLabVIEWと似たように、Windows用のLabVIEWでも、ユーザが作成した内部接続すべてにおいて、1つのLabVIEWインスタンスで有限の1024個のネットワークソケットを使用できるようになりました。この変更は、TCP、UDP、Bluetooth、およびIrDA用のプロトコルVIおよび関数に影響を与えます。これ以外のプロトコル（ネットワークストリーム、ネットワーク共有シェア変数、およびウェブサービス）には影響しません。
- 数学&科学定数およびExpress数学&科学定数の一部に新しい値が追加されました。アボガドロ定数、素電荷、重力定数、モル気体定数、プランク定数、およびリュードベリ定数の値は、CODATA 2014の値と一致するように更新されました。

VI、関数、およびノードについては、『LabVIEWヘルプ』の**目次**タブの**VIと関数のリファレンス**を参照してください。

追加および変更されたクラス、プロパティ、メソッド、およびイベント

LabVIEW 2016では、以下のクラス、プロパティ、メソッドおよびイベントが追加または変更されました。

VIサーバのプロパティとメソッド

LabVIEW 2016では、実行のハイライト（クラス: VI）プロパティが変更されました。このプロパティは、VIの実行のハイライト設定の読み取りおよび書き込みを行うために使用します。このプロパティを使用するには、VIスクリプトを有効にする必要があります。実行のハイライト?（クラス: トップレベルダイアグラム） プロパティとは異なり、再入可能VIのクローンであるVIの実行のハイライト?プロパティを設定できます。

LabVIEW（64ビット）用のアドオン統合

LabVIEW 2016（64ビット）にMicrosoft Office用のLabVIEW Report Generationツールキットが追加されました。

Windows、OS X、およびLinux用の新規64ビットモジュールおよびツールキット

以下のモジュールおよびツールキットの64ビット版が、すべてのプラットフォーム用のLabVIEW 2016（64ビット）でサポートされるようになりました。

- Control Design and Simulationモジュールーシステム同定VI、System Identification Assistant、およびControl Design Assistantを除く
- MathScript RTモジュール
- VI Analyzerツールキット
- **（Windows）** Desktop Execution Traceツールキット
- **（Windows）** Unit Test Frameworkツールキット

システム要件、インストール手順、およびアクティブ化の詳細については、各製品のReadmeを参照してください。

OS X用LabVIEWへの変更

OS X用LabVIEW 2016は、64ビット版のみが提供されています。NIは、OS X用LabVIEWの32ビット版の提供を停止しました。

Linux用LabVIEWへの変更

Linux用LabVIEW 2016（32ビットおよび64ビット）は、64ビット版のLinuxオペレーティングシステムのみをサポートしています。Linux用LabVIEWは、32ビット版のLinuxオペレーティングシステムのサポートを停止しました。

旧バージョンのLabVIEWの機能と変更点

お使いのバージョン以降にリリースされたLabVIEWの各バージョンの新機能を特定するには、それらのバージョンのアップグレードノートを参照してください。これらのドキュメントにアクセスするには、ナショナルインスツルメンツのウェブサイト（`ni.com/jp/info`）で以下のリストから適切なLabVIEWバージョン用のInfo Codeを入力してください。

- LabVIEW 2012アップグレードノートー`upnote12jp`
- LabVIEW 2013アップグレードノートー`upnote13jp`
- LabVIEW 2014アップグレードノートー`upnote14jp`
- LabVIEW 2015アップグレードノートー`upnote15jp`

371780M-0112

2016 年 06 月