

# LabVIEW™ 2019 アップグレード ノート

このアップグレードノートは、Windows、macOS、および Linux 用の LabVIEW を LabVIEW 2019 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べき互換性の問題
- LabVIEW 2019 に新たに追加された機能および動作

## 目次

---

LabVIEW 2019 にアップグレードする.....	2
推奨アップグレードプロセスの概要.....	2
1.VI およびマシン構成のバックアップを作成する.....	3
2.VI の既存の動作をテストして記録する.....	4
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.....	5
4.VI を変換し、動作の変化を検証する.....	5
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.....	8
アップグレードおよび互換性の問題.....	8
LabVIEW 2015 からアップグレードする.....	8
LabVIEW 2016 からアップグレードする.....	8
LabVIEW 2017 からアップグレードする.....	9
LabVIEW 2018 からアップグレードする.....	10
LabVIEW 2019 の機能および変更点.....	13
新しい基礎コレクションタイプ: セットおよびマップ.....	13
新しいショートカットメニュー項目.....	16
アプリケーションビルダの改善点.....	16
フロントパネルの改善点.....	18
プロジェクト項目を置き換える.....	19
コードの一部の実行をハイライト表示する.....	20
履歴プロンプトを使用してデータフローを監視する.....	20
ケースストラクチャのエラー処理の改善.....	21
定数、制御器、および表示器でのテキストのオーバーフローを通知する.....	21

環境の改善点.....	22
追加および変更された VI および関数.....	23
新規プロパティ.....	26
旧バージョンの LabVIEW の機能と変更点.....	26

## LabVIEW 2019 にアップグレードする

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、NI では、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できるよう、より綿密なアップグレードプロセスを推奨しています。



**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、NI Web サイト ([ni.com/info](http://ni.com/info)) で Info Code に「[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

**VI とマシン構成を  
バックアップする**

VI と開発マシンを保護することにより、必要な場合にはファイルを復元してアップグレードプロセスをやり直せるようにしておきます。

**VI の既存の動作を  
テストして記録する**

前バージョンの LabVIEW で VI の標準的動作を把握しておくことにより、アップグレードによって生じる動作の変化をより正確に検知できます。

**LabVIEW、アドオン、  
デバイスドライバを  
インストールする**

すべての NI ソフトウェアを同時にアップグレードすると、アップグレード後の VI は、最新バージョンの LabVIEW 環境に必要なサブ VI、パレット、およびプロパティノードを検出できます。

**VI を変換して、  
動作の変化を検証する**

アップグレード直後に VI を変換してテストすることにより、VI が予想どおりに動作することを確認し、動作が変化した場合でも、実行開始前に修正することができます。



**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、NI Web サイト ([ni.com/info](http://ni.com/info)) で Info Code に「[upgradeOld](#)」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 2019 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ことで、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。

ソースコード管理に VI をサブミットすることにより、VI をバックアップできます。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』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。

### 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 (通常は仮想マシン) を使用し、アップグレードプロセスをテストします。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、NI では、実際の工程を制御または監視するマシンのダウンタイム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推奨しています。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と、LabVIEW の旧バージョンと LabVIEW 2019 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。

- 壊れている VI を特定する一括コンパイルログ

特に複数の人がその 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は、アップグレードする前に VI を一括コンパイルすることが有益です。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』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 を一括コンパイルする**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 のグループがまとめ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することができます。

- 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



**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。

VI テストの詳細については、『LabVIEW ヘルプ』で**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。

## 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。

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 2019 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 2019 の `labview\user.lib` ディレクトリにコピーします。

c. VI パッケージを再インストールする

JKI VI Package Manager (VIPM) を使用して LabVIEW の前のバージョンに VI パッケージをインストールした場合は、VIPM ソフトウェアを使用して LabVIEW 2019 にすべてのパッケージを再インストールしてください。

## 4. VI を変換し、動作の変化を検証する

LabVIEW 2019 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 を特定するために役立つエラーログが作成されます。この情報を「[アップグレードおよび互換性の問題](#)」と併せて使用することにより、LabVIEW 2012 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。



**メモ** NI では、LabVIEW でソース管理を使用して VI をバックアップし、変更をトラッキングすることを推奨しています。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 を前のバージョンに戻すことができます。

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 LabVIEW 2019 すると、VI は LabVIEW 2019 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』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 2019 との違いにより、変更された機能が使用されている VI が壊れる場合があります。壊れた VI を LabVIEW 2019 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. LabVIEW への更新により各 VI が壊れたか確認するには、「[アップグレードおよび互換性の問題](#)」参照してください。

#### c. 動作の変化を特定して修正する

NI 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いますが、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。

- VI Analyzer Upgrade Test を実行する一大規模な 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. 使用している旧バージョン以降の LabVIEW のすべてのバージョンの VI Analyzer Upgrade Test 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、NI Web サイト ([ni.com/info](http://ni.com/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**」を参照して修正します。
- アップグレードドキュメントを読む
    - **アップグレードおよび互換性の問題**—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」トピック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 2019 既知の問題リスト—LabVIEW 2019 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、NI Web サイト ([ni.com/info](http://ni.com/info)) で Info Code に「**lv2019ki**」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 ファイラー—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 2019 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を実行する—旧バージョンの LabVIEW で行ったものと同じテストを LabVIEW 2019 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

以下のアップグレードに関する問題の解決法の詳細については、`labview\manuals` ディレクトリにインストールされている『`troubleshooting_guide.html`』ドキュメント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 を LabVIEW 2019 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。

LabVIEW の新しいバージョンの既知の問題、その他の互換性に関する問題、LabVIEW 2019 の最新の追加機能の詳細については、`labview` ディレクトリにある『`readme.html`』ファイルを参照してください。

### LabVIEW 2015 からアップグレードする

LabVIEW 2015 から LabVIEW 2019 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 からアップグレードする

LabVIEW 2016 から LabVIEW 2019 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。

#### アクターフレームワーク VI の動作の変更

LabVIEW 2016 以前では、ネストされたアクターは、起動前初期化メソッドのエラーが原因で起動できないと、エラーを返し、そのエラーを含んだ最終 Ack メッセージを発呼者アクターに送信します。LabVIEW 2017 以降では、ネストされたアクターは、最終 Ack メッセージを送信せずにエラーを返します。



# LabVIEW 2017 からアップグレードする

LabVIEW2017 から LabVIEW 2019 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 ランタイムエンジンの下位互換性

LabVIEW 2017 以降では、LabVIEW ランタイムエンジンの下位互換性がサポートされています。旧バージョンの LabVIEW で作成されたバイナリおよび VI を、バイナリを再度ビルドしたり、VI を現在のバージョンの LabVIEW で一括コンパイルしたりせずにロードおよび実行できます。たとえば、LabVIEW 2017 以降のバージョンでは、LabVIEW 2017 でビルドされたバイナリおよび VI を再コンパイルせずにロードできます。この改善の対象となるのは、スタンドアロンアプリケーション (EXE)、共有ライブラリ (DLL)、およびバックプロジェクトライブラリです。

バイナリで下位互換性を有効にするには、ビルド仕様に基づいて、特定のダイアログボックスの「上級」ページで以下のチェックボックスをオンにします。

ビルド仕様	ダイアログボックス	チェックボックス
スタンドアロンアプリケーション (EXE)	アプリケーションプロパティ	今後の LabVIEW ランタイムバージョンでこのアプリケーションの実行を許可する
バックプロジェクトライブラリ	バックライブラリプロパティ	今後の LabVIEW バージョンでこのバックライブラリのロードを許可する
共有ライブラリ (DLL)	共有ライブラリプロパティ	今後の LabVIEW バージョンでこの共有ライブラリのロードを許可する

LabVIEW 2017 以降で作成したビルド仕様では、これらのオプションはデフォルトで有効になっています。ビルド仕様を特定のバージョンの LabVIEW に関連付ける場合は、これらのオプションを無効にします。これらのオプションを無効にすると、パフォーマンスプロファイルへの変更を防ぐことができる上、コンパイラのアップグレードによる予期しない問題の発生を回避できます。リアルタイムアプリケーションでは、ダイアログボックスにこれらのオプションは表示されませんが、この機能はデフォルトで有効になっています。

## レポート生成 VI の動作の変更

LabVIEW 2018 では、レポート生成 VI は、標準レポート形式のレポートの生成をサポートしなくなりました。レポートは、HTML、Word、または Excel 形式でのみ生成できません。この動作変更により、次の VI は廃止予定となりました。

- 簡易 VI パネルまたはドキュメントを印刷—この VI は廃止予定となりました。代わりに、「VI パネルまたはドキュメントを印刷」VI を使用してください。
- 簡易テキストレポート—この VI は廃止予定となりました。代わりに、「簡易テキストレポートを作成」VI を使用してください。
- レポートタイプ取得—この VI は廃止予定となりました。代わりに、「レポートタイプ」VI を使用してください。
- 新規レポート—この VI は廃止予定となりました。代わりに、「レポートを作成」VI を使用してください。
- レポートタブ幅設定—この VI は廃止予定となりました。

## 廃止予定の VI、関数、およびノード

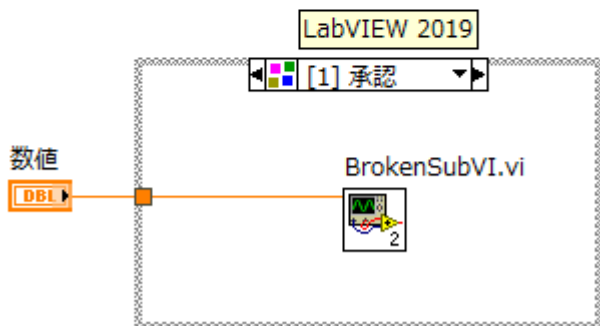
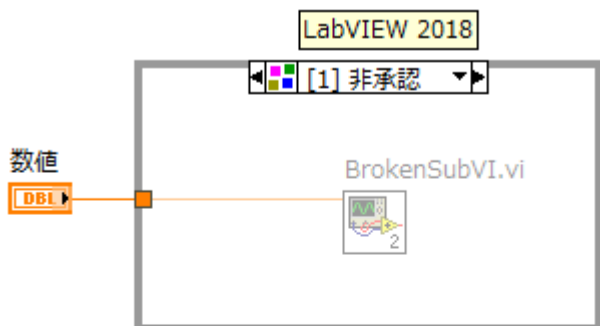
LabVIEW 2018 以降では、「数値を列挙体に変換」VI はサポートされません。代わりに、「タイプに強制変換」関数を使用してください。

# LabVIEW 2018 からアップグレードする

LabVIEW 2018 から LabVIEW 2019 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。

## タイプ特化ストラクチャの動作の変更

LabVIEW 2019 では、タイプ特化ストラクチャがサブダイアグラムを受け入れるか拒否するかを決定する際の構文エラーのチェック動作が変更されました。LabVIEW 2018 では、タイプ特化ストラクチャは、ストラクチャ内のエラー（壊れたワイヤなど）およびサブ VI や他の依存項目からのエラーを、サブダイアグラムを拒否する理由として認識します。LabVIEW 2019 では、タイプ特化ストラクチャは、ストラクチャ内のエラーのみを、サブダイアグラムを拒否する理由として認識します。



タイプ特化ストラクチャの詳細については、『LabVIEW ヘルプ』で **VI と関数のリファレンス**→**プログラミング VI および関数**→**ストラクチャ**→**タイプ特化ストラクチャ**トピックを参照してください。

クラシック、システム、または NXG スタイルでブール制御器および表示器を作成する際の動作の変更

LabVIEW 2019 では、VI がクラシック、システム、または NXG スタイルを使用して制御器および表示器を作成するように構成されている場合に、ブール端子から制御器および表示器を作成する際の動作が変更されました。次の表は、LabVIEW 2018 以前と LabVIEW 2019 で作成されたブール制御器および表示器の外観を比較したものです。

スタイル	タイプ	LabVIEW 2018 以前	LabVIEW 2019
クラシック	制御器		<input checked="" type="checkbox"/> OFF/ON
システム	制御器/表示器	<input type="radio"/> OFF/ON	<input checked="" type="checkbox"/> OFF/ON
NXG	制御器		<input checked="" type="checkbox"/> Off/On



**メモ** 制御器および表示器を作成する際のスタイルを構成するには、**ファイル**→**VI プロパティ** を選択し、**カテゴリ**プルダウンメニューから**編集オプション**を選択した後、**制御器/表示器を作成時の制御器スタイル**リストから適切なスタイルを選択します。

この動作の変更は、次の方法を使用して作成されるブール値の制御器と表示器に適用されます。

- 制御器を作成または表示器を作成メソッドを使用して作成する。
- ブール端子を右クリックし、ショートカットメニューから**作成**→**制御器**または**作成**→**表示器**を選択して作成する。

「データ値リファレンス読み取り/書き込み」境界ノードによる自動エラー処理への動作変更

In Place 要素ストラクチャに 1 組のデータ値リファレンス読み取り/書き込み境界ノードを配置した場合、左右両方の境界ノードに**エラー出力端子**があります。LabVIEW 2018 では、VI で自動エラー処理が有効になっている時にエラーが発生すると、LabVIEW は各未配線の**エラー出力端子**にエラーダイアログボックスを表示します。LabVIEW 2019 では、未配線の**エラー出力端子**の数に関わらず、LabVIEW は 1 つのエラーダイアログボックスのみを表示します。

「データ値リファレンス要素読み取り/書き込み」境界ノードの詳細については、『LabVIEW ヘルプ』で **VI と関数のリファレンス**→**プログラミング VI** および **関数**→**ストラクチャ**→**In Place 要素ストラクチャ**→**データ値リファレンス要素読み取り/書き込み境界ノード**トピックを参照してください。

パスワード保護されたライブラリでのコミュニティメンバーの呼び出しへの動作変更  
LabVIEW 2018 では、フレンド VI がパスワードで保護されているライブラリ内のコミュニティメンバーを呼び出す場合、フレンド VI を初めて編集または実行するときに

ライブラリのパスワードを入力する必要があります。LabVIEW 2019 では、ライブラリのパスワードを入力せずにフレンド VI を編集または実行できます。

定数、制御器、および表示器でのテキストのオーバーフローを通知する

LabVIEW 2019 では、デフォルトで、文字列、数値、タイムスタンプ、テキストリングと列挙体、およびコンボボックスで表示テキストが切れている場合、それが通知されます。この場合、テキストはグレーにフェードアウトされ、矢印が付きます。テキストオーバーフローを無効にするには、制御器、表示器、または定数を右クリックし、**表示項目** → **テキストオーバーフロー** を選択します。


データタイプ解析 VI の端子名の変更

データタイプ解析 VI のトップレベル端子名が、大文字から小文字に変更されました。クラスタ要素など、サブレベルの端子名は変更されていません。

データタイプ解析 VI の詳細については、『LabVIEW ヘルプ』で **VI と関数のリファレンス** → **プログラミング VI および関数** → **クラスタ、クラス、バリエーション VI および関数** → **バリエーション VI および関数** → **データタイプ解析 VI** ブックを参照してください。

## LabVIEW 2019 の機能および変更点

---

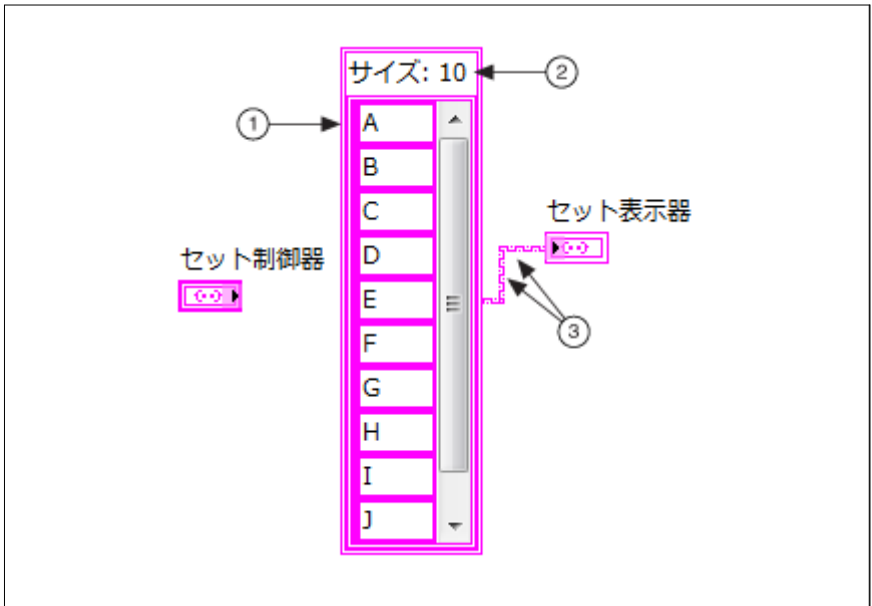
意見交換アイコン  は、[ni.com](https://ni.com) の [NI Idea Exchange](https://ni.com/ideaexchange) ディスカッションフォーラム (英語) での製品に関する提案を元に開発された新機能を示しています。

LabVIEW の各バージョン特有のアップグレードおよび互換性の問題については、「[アップグレードおよび互換性の問題](#)」を参照してください。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 2019 における追加機能については、`labview\readme` ディレクトリにある『`readme.html`』ファイルを参照してください。

## 新しい基礎コレクションタイプ: セットおよびマップ

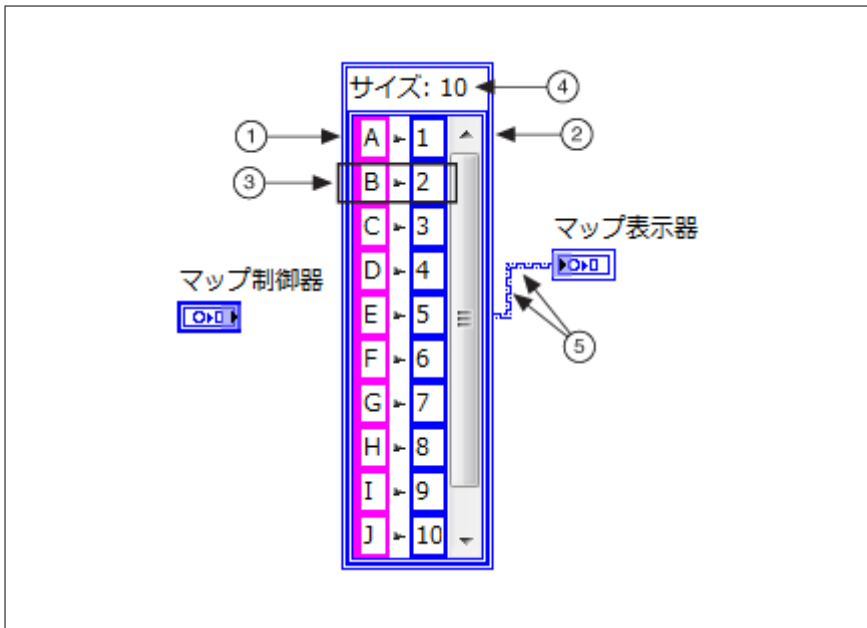
LabVIEW 2019 には、同種のデータのコレクションを集約するためのコレクションデータタイプであるセットとマップが追加されました。両タイプとも一意の要素またはエントリをソートされた順序で維持するため、データサイズが大きい場合でも、配列などのソートされていないデータ構造よりもデータの検索、挿入、変更、および削除を迅速に実行できます。

- **セット**—同じデータタイプの一意的要素のコレクションです。



①	要素
②	サイズ—セット内の要素の数
③	セットのワイヤーワイヤの色は、要素のデータタイプの色と一致する

- マップ**—キーと値のペアから構成されるエントリのコレクションです (キーはそれぞれ一意)。キーと値はそれぞれ任意のデータタイプにできます。マップは、キーを使用して値のルックアップを行うため、ディクショナリと呼ばれることもあります。



①	キー
②	値
③	エントリー関連付けられたキーと値のペア
④	サイズ—マップ内の要素の数
⑤	マップのワイヤーワイヤの色は、値のデータタイプの色と一致する

**プログラミング→コレクション** パレットのセットおよびマップ VI と関数、および **データコンテナ** パレットのセットおよびマップ制御器と表示器を使用して、セットとマップを作成して操作します。

**メモ** セットやマップ内の埋め込みデータは、対話的またはプログラムのに変更できません。セットまたはマップデータは、セットまたはマップのフロントパネル端子に書き込むか、セットまたはマップを右クリックして**データ操作→データをコピー**または**データ操作→データを貼り付け**を選択するなどの操作により、全体として更新する必要があります。

セットおよびマップの詳細については、『LabVIEW ヘルプ』で**基本機能→文字列、クラスタ、配列、およびコレクション**を使用して**データをグループ化する→概念→コレクション**を使用して**データをグループ化する**トピックを参照してください。





セットおよびマップ VI および関数の詳細については、『LabVIEW ヘルプ』で **VI と関数のリファレンス→プログラミング VI および関数→コレクション VI および関数ブック** を参照してください。

セットまたはマップを使用したデータ操作のサンプルについては、次の VI およびプロジェクトを参照してください。

- labview\examples\Collections\Set Collection - Word Counting.vi
- labview\examples\Collections\Map Collection - Comparing Test Results.vi
- labview\examples\Collections\Map Collection - Word Counting.vi
- labview\examples\Design Patterns\Registration Map\Registration Map Usage.lvproj

## 新しいショートカットメニュー項目

LabVIEW 2019 には、以下のショートカットメニューが新たに追加されました。

- **定数を作成、制御器を作成、表示器を作成**—端子から定数、制御器、表示器を作成します。これらのショートカットメニュー項目は、以前から使用できましたが、ショートカットメニューの一番上に重複して表示されるようになりました。
-  **変換を置換**—数値変換関数を右クリックして**変換を置換**を選択すると、ある数値変換関数から別の数値変換関数に変換されます。(NI ディスカッションフォーラムメンバー EthanStern の提案による)
-  **カラーボックスを作成**—カラー端子を右クリックして**定数を作成、制御器を作成、または表示器を作成**を選択すると、カラーボックス定数、制御器、または表示器が作成されます。(NI ディスカッションフォーラムメンバー altenbach の提案による)
-  **選択からクラスタを作成**—定数、制御器、または表示器を選択して右クリックして**作成→選択からのクラスタ**を選択すると、選択された定数、制御器、または表示器からクラスタが作成されます。(NI ディスカッションフォーラムメンバー okubik の提案による)
-  **スカラ定数または制御器を作成**—配列端子を右クリックして**作成→スカラ→定数または作成→スカラ→制御器**を選択すると、スカラ定数または制御器が作成されます。(NI ディスカッションフォーラムメンバー altenbach の提案による)
- **比較を置換**—比較関数を右クリックして**比較を置換**を選択すると、ある比較関数から別の比較関数に変換されます。

## アプリケーションビルダの改善点

LabVIEW 2019 では、LabVIEW アプリケーションビルダおよびビルド仕様が以下のように改善されました。

(Windows) アプリケーションをパッケージインストーラとして配布する

パッケージインストーラ(.exe)を作成すると、NI パッケージマネージャを介してクライアントにアプリケーションを配布できます。パッケージインストーラには、クラ



クライアントがネットワークアクセスなしでもパッケージをインストールできるように、すべてのパッケージ依存項目が含まれています。

パッケージインストーラを作成するには、**ビルド仕様**を右クリックし、**新規→パッケージ**を選択し、**パッケージプロパティ**ダイアログボックスの**パッケージインストーラ**ページで**パッケージインストーラを作成する**を選択します。

NI パッケージをフィードにパブリッシュする

NI パッケージをフィードにパブリッシュすると、クライアントは、そのフィードをサブスクライブして更新通知を受信し、ネットワーク経由で NI パッケージをインストールできます。

フィードとは、含まれているパッケージに関する情報が記載されたマニフェストが含まれているパッケージファイルの集合です。相互に依存関係のある複数のパッケージを配布する場合、NI ではフィードを使用することを推奨しています。フィードを使用すると、エンドユーザのアクセスを目的としたすべての NI パッケージをホストおよび維持する単一の場所を作成できます。

LabVIEW 2019 では、「**パッケージプロパティ**」ダイアログボックスの**フィード**ページにある以下のオプションを使用して、NI パッケージをローカルフィードに追加するか、SystemLink フィードにパブリッシュできます。

- **フィードに追加**—フィードに NI パッケージを追加します。クライアントは、NI パッケージマネージャを使用してフィードをサブスクライブし、パッケージをインストールします。
- **SystemLink フィードにパブリッシュする**—LabVIEW で作成した NI パッケージを SystemLink サーバ上のフィードにパブリッシュします。クライアントは、SystemLink で NI パッケージを検索してインストールできます。

パッケージ、パッケージインストーラ、およびフィードの概念については、『LabVIEW ヘルプ』で**基礎機能→アプリケーションの作成および配布→ビルド仕様を作成する→配布のタイプ**トピックを参照してください。

パッケージまたはパッケージインストーラの作成してクライアントに配布する方法については、『LabVIEW ヘルプ』で**基礎機能→アプリケーションの作成および配布→ビルド仕様を作成する→配布用のパッケージを作成する**トピックを参照してください。

アプリケーションビルダのその他の改善点

「**パッケージプロパティ**」ダイアログボックスの**パッケージ属性**ページが**パッケージ**と名前変更されました。**パッケージ**ページに以下のオプションが新たに追加されました。

- **NI 証明書パッケージを含める**—クライアントが NI パッケージマネージャでアプリケーションをインストールする際に、NI 証明書パッケージ (ni-certificates) を推奨される依存項目として含めることを指定します。
- **ランタイム展開パッケージリストに表示する**—**フィルタオプション**で**ランタイム展開パッケージのみ**を選択した場合に、**依存項目**ページの**関連パッケージ**リストにパッケージを表示することを指定します。
- **製品によりフィルタした場合に NI パッケージマネージャに表示する**—NI パッケージマネージャで**製品のみ**を選択してパッケージをフィルタ処理した場合に、このパッケージを表示することを指定します。作成するパッケージが、NI パッケージマネージャの**製品を参照**セクションに表示したい製品である場合は、このオプションを有効にします。

## フロントパネルの改善点

LabVIEW 2019 では、フロントパネルが以下のように改善されました。

セットおよびマップ制御器と表示器

LabVIEW 2019 では、**配列**、**行列&クラスタ**パレットが**データコンテナ**に名前変更されました。このパレットには、マップおよびセット制御器と表示器が新たに追加されました。

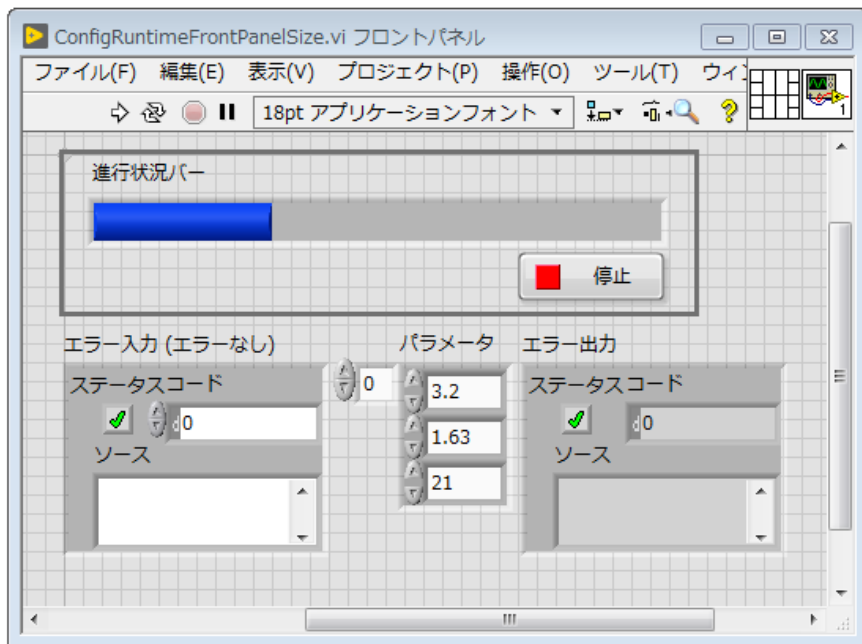
データコンテナ制御器および表示器の詳細については、『LabVIEW ヘルプ』で**基礎機能**→**フロントパネルを作成する**→**概念**→**フロントパネルの制御器および表示器**→**データコンテナ制御器および表示器**トピックを参照してください。

NXG スタイルの制御器と表示器の追加


LabVIEW 2019 では、「NXG スタイル」パレットに制御器および表示器がさらに追加され、LabVIEW NXG と同じ表示スタイルの数値、ブール、コンテナ、および I/O 名制御器や表示器をより簡単に作成できるようになりました。

編集モードで実行時のフロントパネル境界を表示する

🗨️ LabVIEW 2019 では、実行時のフロントパネルのサイズとスクロール位置を記憶するように VI を構成できます。VI が編集モードのとき、LabVIEW はその設定をフロントパネル上で長方形の枠線として表示します。この長方形の枠線は、VI の実行中は表示されません。以下の図では、長方形の枠線が、VI の実行時に表示されるフロントパネルの部分の位置とサイズを示しています。



実行時のフロントパネルの境界を表示する方法については、『LabVIEW ヘルプ』で**基礎機能**→**フロントパネルを作成する**→**操作手順**→**ユーザインタフェースを設計する**→**編集モードで実行時のフロントパネル境界を表示する**トピックを参照してください。

 **メモ** LabVIEW では、単一ページのフロントパネルに対してのみ実行時のフロントパネル境界を表示できます。

(NI ディスカッションフォーラムメンバー attenbach の提案による)


## プロジェクト項目を置き換える

LabVIEW 2019 では、**バックライブラリ**で**置換**ショートカットメニューが**置換...**に名前変更されました。LabVIEW プロジェクトの VI、クラス、またはライブラリは、同じタイプの別の項目に置き換えることができます。**プロジェクトエクスプローラ**ウィンドウでプロジェクト項目を右クリックして**置換...**を選択し、元の項目を置き換えるファイルを選択します。

プロジェクト項目の置き換えの詳細については、『LabVIEW ヘルプ』で**基礎機能**→**プロジェクトとターゲットを操作する**→**概念**→**LabVIEW でプロジェクトを管理する**→**プロジェクト項目を置き換える**トピックを参照してください。

## コードの一部の実行をハイライト表示する

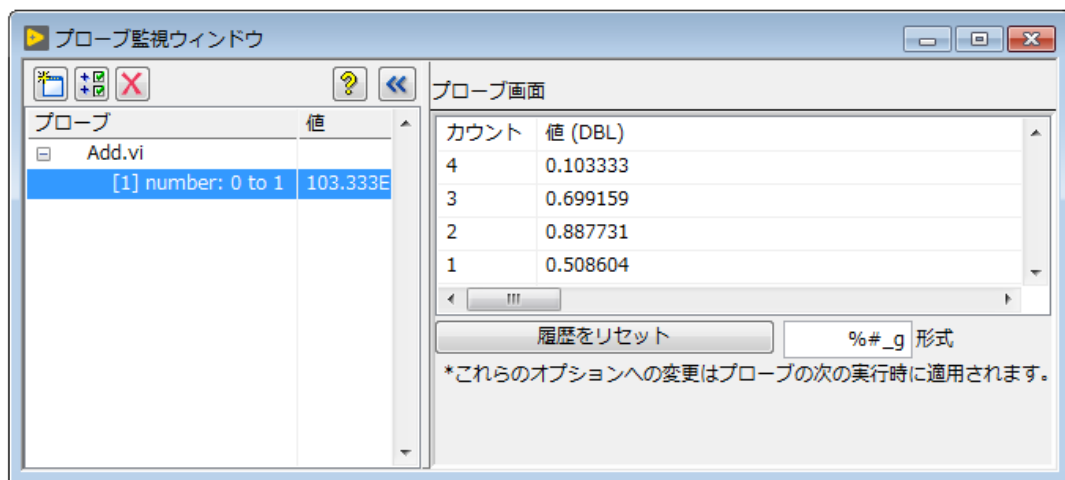
LabVIEW 2019 では、VI のブロックダイアグラム全体でなく、コードの一部の実行をハイライト表示できます。ワイヤを右クリックして**カスタムプローブ**→**実行のハイライトをトグル**を選択すると、実行のハイライトを切り替えるポイントを設定できます。VI の実行中、このプローブは、指定されたポイントで実行のハイライトを切り替えます。

 **メモ** 実行のハイライトをトグルカスタムプローブは、RT ターゲットではサポートされていません。

(NI ディスカッションフォーラムメンバー therealkilkenny の提案による)

## 履歴プローブを使用してデータフローを監視する

LabVIEW 2019 では、ワイヤ上に履歴プローブを配置することにより、ワイヤを流れたデータの履歴を表示できます。



履歴プローブは、以下のデータタイプに対して使用できます。

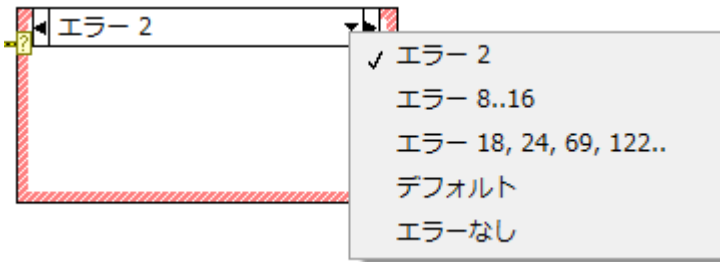
- ブール
- 文字列
- パス
- I8
- I16
- I32
- I64
- U8

- U16
- U32
- U64
- SGL
- DBL
- EXT
- CSG
- CDB
- CXT

履歴プローブをワイヤ上に配置するには、ワイヤを右クリックして**カスタムプローブ**を選択し、データタイプの履歴プローブオプションを選択します。

## ケースストラクチャのエラー処理の改善

🗨️ LabVIEW 2019 では、数値または範囲を入力する場合と同じ構文を使用してセレクトラベルにエラー値または範囲を入力することにより、ケースストラクチャが特定のエラーやエラーリストに対して特定のサブダイアグラムを実行するように構成できます。

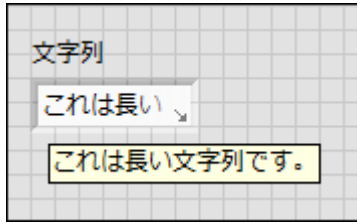


(NI ディスカッションフォーラムメンバー Hueter の提案による)

ケースストラクチャでのエラー処理の詳細については、『LabVIEW ヘルプ』で**基礎機能**→VI を実行およびデバッグする→**操作手順**→**エラーチェックとエラー処理**→**ケースストラクチャを使用してエラーを処理する**トピックを参照してください。

## 定数、制御器、および表示器でのテキストのオーバーフローを通知する

🗨️ LabVIEW 2019 では、文字列、数値、タイムスタンプ、テキストリングと列挙体、およびコンボボックスで、表示テキストが収まりきらずに切れている場合、テキストがフェードアウトして矢印が表示されます。矢印の上にカーソルを置くと、ヒントラベルにすべてのテキストが表示されます。




LabVIEW 2019 では、デフォルトでテキストオーバーフローが有効になっています。テキストオーバーフローを無効にするには、制御器、表示器、または定数を右クリックし、**表示項目** → **テキストオーバーフロー** を選択解除します。

(NI ディスカッションフォーラムメンバー Lavezza の提案による)

## 環境の改善点

LabVIEW 2019 では、LabVIEW 環境が以下のように改善されました。

- **ダイナミックディスパッチサブ VI ノード** をダブルクリックすると、「**実装を選択**」ダイアログボックスに、選択された VI のブロックダイアグラムのプレビューを表示する **ダイアグラムプレビュー** というセクションが表示されるようになりました。(NI ディスカッションフォーラムメンバー PrimaryKey の提案による)
- (Windows) LabVIEW でインストーラをビルドした場合の実行ファイルのデフォルト名が、`setup.exe` から `install.exe` に変更されました。この動作の変更は、`setup.exe` という名前を使用しているビルド後のプロセスに影響を与える可能性があります。実行ファイルに別の名前を指定するには、**インストーラプロパティ** ダイアログボックスの **製品情報** ページにある **インストーラ名** オプションを使用します。
- エラーリングのドロップダウン矢印を選択すると、「**エラーを選択**」ダイアログボックスに、指定されたエラーコードの範囲内でキーワード検索を行える **フィルタ** オプションが表示されるようになりました。このオプションは、**エラーリスト** 表に表示するエラーコードのリストを絞り込む場合に使用します。

-  **プロパティダイアログボックスの外観** ページで、フロントパネルまたはブロックダイアグラムの左座標と上座標の位置を設定できます。(NI ディスカッション フォーラムメンバー smmarlow の提案による)
- **クイックドロップ** キーボードショートカット<Ctrl-F>を使用し、フロントパネルとブロックダイアグラムのオブジェクトと配置を調整します。<Ctrl-Space>を押してから<Ctrl-F>を押すと、LabVIEW は以下のクリーンアップ動作を実行します。



**メモ** <Ctrl-U>を押すと、フロントパネルとブロックダイアグラム上のオブジェクトを整理することもできます。

- フロントパネルでは、制御器と表示器をコネクタペーンの配置と一致するように配置し、フロントパネルをサイズ変更してプライマリモニタの左上の一定の位置に移動します。
- ブロックダイアグラムでは、一番上および/または一番左にあるブロックダイアグラムオブジェクトを基準としてブロックダイアグラムを適切な位置にスクロールし、ブロックダイアグラムのサイズを変更してプライマリモニタの左上の一定の位置に移動します。

## 追加および変更された VI および関数

LabVIEW 2019 には、以下の VI および関数が新たに追加されました。VI、関数、およびノードについては、『LabVIEW ヘルプ』の「**VI と関数のリファレンス**」ブックを参照してください。

### 新しい VI および関数

マップとセット VI および関数

「プログラミング」パレットに、以下のマップおよびセット VI と関数を含む「コレクション」サブパレットが新たに追加されました。

- マップ VI および関数—マップを作成および操作します。
  - マップを作成
  - コレクションサイズ
  - マップを配列に変換
  - 空のコレクション?
  - In Place マップアクセス
  - マップに挿入
  - マップで検索
  - マップ定数
  - マップの最大&最小キーを読み取る

- マップから削除
- 登録マップ VI
  - 登録マップ: 登録を確認
  - 登録マップ: 登録
  - 登録マップ: 登録解除
- セット VI および関数—マップを作成および操作します。
  - セットを作成
  - コレクションサイズ
  - セットを配列に変換
  - 空のコレクション?
  - セットに挿入
  - セットの要素?
  - セットの最大&最小を読み取る
  - セットから削除
  - セット直積
  - セット定数
  - セット差集合
  - セット共通集合
  - セット対称差
  - セット和集合

#### JSONtext VI

「文字列を平坦化/非平坦化」パレットに、LabVIEW で JSON をプログラミングするための JSONtext VI をインストールするリンクが追加されました。「文字列を平坦化/非平坦化」パレットで「**JSONtext アドオンをインストール**」をクリックすると、JKI VI Package Manager (VIPM) から JSONtext アドオンをインストールできます。JSONtext VI は、**アドオン→JSONtext** パレットに表示されます。

(JSONtext アドオンの作者である Dr. James David Powell に特に感謝いたします)

#### データタイプ解析 VI

「データタイプ解析」パレットに以下の VI が新たに追加されました。

- マップコレクション情報を取得—入力バリエーションデータに格納されているデータタイプからマップ情報を取得します。
- セットコレクション情報を取得—入力バリエーションデータに格納されているデータタイプからセット情報を取得します。
- エラークラスター—入力バリエーションに格納されたデータタイプがエラークラスターかどうかを示します。




## 上級ファイル VI

「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。

- 再帰ディレクトリを作成—ディレクトリとそれが必要とする親ディレクトリを作成します。
- ファイルとフォルダを作成—指定されたパスに、ファイルとそのパス内の存在しないフォルダをすべて作成します。

## 数値 VI および関数

「数値」パレットに、以下の定数および VI が追加されました。

-  非定数一値 NaN をブロックダイアグラムに返します。(NI ディスカッションフォーラムメンバー attenbach の提案による)
- 乱数 (範囲)—指定された範囲から乱数値を生成します。この VI は、U64、I64、および DBL データタイプで使用できます。

## その他の文字列 VI および関数

「その他の文字列」パレットに以下の VI が新たに追加されました。

- 1D 文字列配列をデリミタ文字列に変換—1D 文字列配列の要素をデリミタ文字で区切られた入力配列要素を含む単一文字列に変換します。
- デリミタ文字列を 1D 文字列配列に変換—デリミタ文字列のサブ文字列を 1D 配列文字列配列の要素に変換します。

## その他の新しい VI および関数

LabVIEW 2019 には、以下のその他の VI および関数が新たに追加されました。

- 「配列」パレットに、「1D 配列から重複を削除」VI が追加されました。この VI は、1D 配列から重複した要素を削除します。この VI は要素の元の順序を保持します。
- 「タイプをアサート」パレットに「ストラクチャタイプの不一致をアサート」関数が新たに追加されました。この関数は、入力タイプが、指定された不一致入力のいずれかのデータタイプと同じ場合、タイプ定義とタイプ名を無視して発呼者 VI を壊します。「ストラクチャタイプの不一致をアサート」をタイプ特化ストラクチャと併せて使用すると、特定のデータタイプ用に順応性 VI (`.vim`) のコードの一部をカスタマイズしたり、順応性 VI が特定のデータタイプを拒否するように強制したりできます。
- 「プロトコル」パレットに「構成されたネットワークを待機」VI が追加されました。この VI は、システムがリモートホストと接続できるまで待機するために使用します。
- 「同期」パレットに「データフローを同期」VI が追加されました。この VI は、先行コードの実行後に入力ワイヤの値を渡します。この VI を使用すると、複数の並

列コードのパスをデータフローの単一ポイントで同期させることにより、特定の順序での実行を保証できます。

- 「アプリケーション制御」パレットに、「コマンドライン引数を取得」VIが追加されました。このVIは、LabVIEW または LabVIEW でビルドされたアプリケーションが起動したときにコマンドラインから渡されたユーザ定義の引数を返します。ユーザ定義の引数は、コマンドラインでスペースに囲まれた2つのハイフン (-) の後から始まります。

## 変更された VI および関数

LabVIEW 2019 では、以下の関数に変更されました。

- **接続**→Python パレットにある Python ノードが、新たにブールデータタイプをサポートするようになりました。Python ノードでは、数値配列を NumPy 配列にマーシャリングすることもできます。

## 新規プロパティ

LabVIEW 2019 では、以下のプロパティに変更されました。

- VI クラスに、**ブロックダイアグラムウィンドウ:アライメントグリッドサイズ**および**フロントパネルウィンドウ:アライメントグリッドサイズ**の各プロパティが追加されました。これらのプロパティは、VI のブロックダイアグラムまたはフロントパネルのアライメントグリッドサイズを読み書きするために使用します。これら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。

## 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、NI Web サイト ([ni.com/info](http://ni.com/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 2015 アップグレードノート—[upnote15jp](#)
- LabVIEW 2016 アップグレードノート—[upnote16jp](#)
- LabVIEW 2017 アップグレードノート—[upnote17jp](#)
- LabVIEW 2018 アップグレードノート—[upnote18jp](#)

情報は事前の通知なしに変更されることがあります。NI の商標の詳細については、[ni.com/trademarks](http://ni.com/trademarks) の NI Trademarks and Logo Guidelines (英語) を参照してください。本書中に記載されたその他の製品名及び企業名は、それぞれの企業の商標又は商号です。NI の製品及び技術を保護する特許については、ソフトウェアで参照できる特許情報 (ヘルプ>特許)、メディアに含まれている patents.txt ファイル、又は [ni.com/patents](http://ni.com/patents) からアクセスできる National Instruments Patent Notice のうち、該当するリソースから参照してください。エンドユーザ使用許諾契約 (EULA) 及び他社製品の法的注意事項はご使用の NI 製品の Readme ファイルにあります。NI の輸出関連法規遵守に対する方針については、また必要な HTS コード、ECCN (Export Control Classification Number)、その他の輸出入に関する情報の取得方法については、「輸出関連法規の遵守に関する情報」([ni.com/legal/ja/export-compliance](http://ni.com/legal/ja/export-compliance)) を参照してください。NI は、本書に記載の情報の正確性について、一切の明示又は黙示の保証を行わず、技術的な誤りについて一切の責任を負いません。米国政府のお客様へ: 本書に含まれているデータは、民間企業の費用により作成されており、民間機関用の連邦調達規則 52.227-14 と軍事機関用の国防省連邦調達規則補足 252.227-7014 および 252.227-7015 に基づく限定権利及び制約付データ権利の条項の適用を受けます。

© 1998–2019 National Instruments. All rights reserved.

371780R-0112 2019年3月7日