



Informatica® Cloud Data Integration
October 2022

タスクフロー

© 著作権 Informatica LLC 2006, 2022

本ソフトウェアおよびマニュアルは、使用および開示の制限を定めた個別の使用許諾契約のもとでのみ提供されています。本マニュアルのいかなる部分も、いかなる手段（電子的複写、写真複写、録音など）によっても、Informatica LLC の事前の承諾なしに複製または転載することは禁じられています。

米政府の権利プログラム、ソフトウェア、データベース、および関連文書や技術データは、米国政府の顧客に配信され、「商用コンピュータソフトウェア」または「商業技術データ」は、該当する連邦政府の取得規制と代理店固有の補足規定に基づきます。このように、使用、複製、開示、変更、および適応は、適用される政府の契約に規定されている制限およびライセンス条項に従うものとし、政府契約の条項によって適当な範囲において、FAR 52.227-19、商用コンピュータソフトウェアライセンスの追加権利を規定します。

Informatica、Informatica Cloud、Informatica Intelligent Cloud Services、PowerCenter、PowerExchange、および Informatica ロゴは、米国およびその他の国における Informatica LLC の商標または登録商標です。Informatica の商標の最新リストは、Web (<https://www.informatica.com/trademarks.html>) にあります。その他の企業名および製品名は、それぞれの企業の商標または登録商標です。

本ソフトウェアまたはドキュメンテーション（あるいはその両方）の一部は、第三者が保有する著作権の対象となります。必要な第三者の通知は、製品に含まれています。

本マニュアルの情報は、予告なしに変更されることがあります。このドキュメントで問題が見つかった場合は、infa_documentation@informatica.com までご報告ください。

Informatica 製品は、それらが提供される契約の条件に従って保証されます。Informatica は、商品性、特定目的への適合性、非侵害性の保証等を含めて、明示的または黙示的ないかなる種類の保証をせず、本マニュアルの情報を「現状のまま」提供するものとします。

発行日: 2022-12-14

目次

序文	6
Informatica のリソース	6
Informatica マニュアル	6
Informatica Intelligent Cloud Services Web サイト	6
Informatica Intelligent Cloud Services コミュニティ	6
Informatica Intelligent Cloud Services マーケットプレイス	7
データ統合コネクタのドキュメント	7
Informatica ナレッジベース	7
Informatica Intelligent Cloud Services Trust Center	7
Informatica グローバルカスタマサポート	7
第 1 章 : タスクフローとリニアタスクフロー	8
第 2 章 : タスクフロー	9
タスクフローのステップ	9
タスクフローの作成	11
タスクフローテンプレート	11
基本	12
並列タスク	12
ディシジョンがある並列タスク	12
シーケンシャルタスク	13
ディシジョンがあるシーケンシャルタスク	13
単一タスク	13
タスクフロープロパティの設定	13
全般的なプロパティ	13
開始プロパティ	14
入力フィールド	17
出力フィールド	19
一時フィールド	20
詳細プロパティ	21
メモ	22
タスクフローステップのプロパティの設定	22
割り当てステップ	22
データタスクステップ	23
通知タスクステップ	28
コマンドタスクステップ	33
File Watch タスクステップ	40
取り込みタスクステップ	42
サブタスクフローステップ	45
決定ステップ	46

並列パスステップ.	48
ジャンプステップ.	48
終了ステップ.	49
待機ステップ.	49
スローステップ.	49
ランタイムパラメータ.	52
タスクフローのパラメータ.	53
タスクフローでのパラメータの上書き.	54
タスクフローでのパラメータ使用のガイドラインおよびベストプラクティス.	55
例: データタスクステップを使用したパラメータの上書き.	56
パラメータセット.	57
パラメータセットの要件.	57
パラメータスコープ.	59
パラメータセットのサンプル.	61
パラメータセットに関するルールおよびガイドライン.	61
式エディタ.	62
ヒント: XQuery 3.0 を使用した式の作成.	63
キーボードショートカット.	65
タスクフロー関数.	65
addToDate.	66
base64Decode.	69
dateDiff.	69
decode.	72
getAssetLocation.	75
getAssetName.	75
getDatePart.	75
getInstanceStartTime.	77
iif.	78
in.	79
instr.	80
isNull.	83
lastDay.	84
lpad.	85
ltrim.	87
round (Numbers).	88
rtrim.	90
toChar (Numbers).	92
toDate.	94
toDecimal.	96
toInteger.	98
trunc.	99
[検証] パネルの使用.	102

タスクフローの実行.	103
タスクフローデザイナーからのタスクフローの実行.	104
タスクフローの実行タスクフロー入力の使用.	104
タスクフローの作成入力.	104
タスクフロー入力によるタスクフローの実行.	106
タスクフロー入力の削除.	107
タスクフローのパブリッシュ.	108
タスクフローの一括パブリッシュ.	109
API としてのタスクフローの実行.	110
ブラウザ経由での入力の受け渡し.	111
REST クライアント経由での入力の受け渡し.	111
一時停止したタスクフローの再開.	113
パラメータセットを使用したタスクフローの実行.	114
タスクフローの実行 RunAJob ユーティリティの使用.	117
コネクタファイルリスナ経由のタスクフローの呼び出し.	117
タスクフローのスケジュール.	118
ステータスリソースを使用したタスクフローステータスの監視.	119
タスクフローの例.	126
第 3 章 : リニアタスクフロー.	128
リニアタスクフロージョブのスケジュール設定.	129
リニアタスクフローの設定.	129
リニアタスクフローの実行.	131
リニアタスクフローまたはサブタスクの停止.	131
索引.	132

序文

「タスクフロー」を使用して、指定した時間に特定の順序でデータ統合タスクを実行するために、動的なリニアタスクフローをセットアップする方法について学びます。

Informatica のリソース

Informatica は、Informatica Network やその他のオンラインポータルを通じてさまざまな製品リソースを提供しています。リソースを使用して Informatica 製品とソリューションを最大限に活用し、その他の Informatica ユーザーや各分野の専門家から知見を得ることができます。

Informatica マニュアル

Informatica マニュアルポータルでは、最新および最近の製品リリースに関するドキュメントの膨大なライブラリを参照できます。マニュアルポータルを利用するには、<https://docs.informatica.com> にアクセスしてください。

製品マニュアルに関する質問、コメント、ご意見については、Informatica マニュアルチーム (infa_documentation@informatica.com) までご連絡ください。

Informatica Intelligent Cloud Services Web サイト

Informatica Intelligent Cloud Services Web サイト (<http://www.informatica.com/cloud>) にアクセスできます。このサイトには、Informatica Cloud 統合サービスに関する情報が含まれます。

Informatica Intelligent Cloud Services コミュニティ

Informatica Intelligent Cloud Services コミュニティを使用して、技術的な問題について議論し、解決します。また、技術的なヒント、マニュアルの更新情報、FAQ（よくある質問）への答えを得ることもできます。

次の Informatica Intelligent Cloud Services コミュニティにアクセスします。

<https://network.informatica.com/community/informatica-network/products/cloud-integration>

開発者は、次の Cloud 開発者コミュニティで詳細情報を確認したり、ヒントを共有したりできます。

<https://network.informatica.com/community/informatica-network/products/cloud-integration/cloud-developers>

Informatica Intelligent Cloud Services マーケットプレイス

Informatica マーケットプレイスにアクセスすると、データ統合コネクタ、テンプレート、およびマップレットを試用したり購入したりできます。

<https://marketplace.informatica.com/>

データ統合コネクタのドキュメント

データ統合コネクタのドキュメントには、マニュアルポータルからアクセスできます。マニュアルポータルを利用するには、<https://docs.informatica.com> にアクセスしてください。

Informatica ナレッジベース

Informatica ナレッジベースを使用して、ハウツー記事、ベストプラクティス、よくある質問に対する回答など、製品リソースを見つけることができます。

ナレッジベースを検索するには、<https://search.informatica.com> にアクセスしてください。ナレッジベースに関する質問、コメント、ご意見の連絡先は、Informatica ナレッジベースチーム (KB_Feedback@informatica.com) です。

Informatica Intelligent Cloud Services Trust Center

Informatica Intelligent Cloud Services Trust Center は、Informatica のセキュリティポリシーおよびリアルタイムでのシステムの可用性について情報を提供します。

Trust Center (<https://www.informatica.com/trust-center.html>) にアクセスします。

Informatica Intelligent Cloud Services Trust Center にサブスクライブして、アップグレード、メンテナンス、およびインシデントの通知を受信します。[Informatica Intelligent Cloud Services Status](#) ページには、すべての Informatica Cloud 製品の実稼働ステータスが表示されます。メンテナンスの更新はすべてこのページに送信され、停止中は最新の情報が表示されます。更新と停止の通知がされるようにするには、Informatica Intelligent Cloud Services の 1 つのコンポーネントまたはすべてのコンポーネントについて更新の受信をサブスクライブします。すべてのコンポーネントにサブスクライブするのが、更新を逃さないようにするための最良の方法です。

登録するには、<https://status.informatica.com/> に移動し、**[更新を購読登録]** をクリックします。その後、電子メール、SMS テキストメッセージ、Webhook、RSS フィードとして、またはこの 4 つを任意に組み合わせて送信された通知を受信することを選択ができます。

Informatica グローバルカスタマサポート

電話またはオンラインでカスタマサポートセンターに連絡できます。

オンラインサポートについては、Informatica Intelligent Cloud Services の **[サポート要求の送信]** をクリックしてください。またオンラインサポートを使用して問題を記録することもできます。オンラインサポートを利用するには、ログインが必要です。<https://network.informatica.com/welcome> でログイン要求できます。

Informatica グローバルカスタマサポートの電話番号は、Informatica の Web サイト <https://www.informatica.com/services-and-training/support-services/contact-us.html> に掲載されています。

第 1 章

タスクフローとリニアタスクフロー

タスクフローまたはリニアタスクフローを作成できます。

タスクフロー

タスクフローを使用して、データ統合タスクの実行シーケンスを制御します。タスクを並列に実行し、拡張意思決定基準を使用したり、タスクの時間を計測したり、他の拡張オーケストレーションを実行したりできます。

リニアタスクフロー

リニアタスクフローは指定した順序で複数のタスクを順番に実行します。

第 2 章

タスクフロー

タスクフローを使用して、以前のタスクの出力に基づき、データ転送タスク、動的マッピングタスク、マッピングタスク、PowerCenter タスク、または同期タスクの実行順序を制御します。

例えば、2 つのマッピングタスクを順番に実行するとします。ただし、最初のマッピングタスクの終了後には警告が表示されても、2 番目のマッピングタスクをすぐに実行しないとします。その代わりに、2 番目のマッピングタスクは 2 時間後に実行するとします。このシナリオを調整するためにタスクフローを作成できます。

注: タスクフローを作成する前に、使用するタスクを準備しておく必要があります。タスクフロー作成プロセスでタスクを作成することはできません。

API として、またはコネクタファイルリサナ経由で、タスクフローデザイナーからタスクフローを呼び出して実行できます。スケジュールに従ってタスクフローを実行する事もできます。

タスクフローのステップ

タスクフローのステップを使用して、データ統合タスクを追加および調整します。さまざまなタイプのステップをタスクフローに追加できます。タスクフローにステップを追加するには、左側のパレットからステップをドラッグします。

次の手順をタスクフローに追加できます。

割り当て

「割り当て」ステップを使用して、フィールドの値を設定します。フィールドは、タスクフローに関連するデータを処理するデータホルダです。入力フィールドと一時フィールドを使用して、フィールドの値を設定できます。

入力フィールドでは、タスクフローの実行に入力ができます。タスクフローは、データを内部的に処理するために一時フィールドを使用します。

データタスク

データタスクステップを使用して、データ転送タスク、動的マッピングタスク、マッピングタスク、PowerCenter タスク、または同期タスクをワークフローに追加します。タスクフローによるエラーおよび警告の処理、スケジュールに基づいたアクションの実行、およびランタイムパラメータの上書きの方法については、設定することができます。

通知タスク

指定した受信者に通知を送信するには、通知タスクステップを使用します。

電子メール通知を送信する通知タスクステップを設定できます。例えば、タスクフローのデータタスクステップで見つかった成功した行とエラー行の数について受信者に知らせる電子メール通知を送信できます。

コマンドタスク

コマンドタスクステップを使用して、Secure Agent マシン上の複数のスクリプトファイルからシェルスクリプトまたはバッチコマンドを実行します。例えば、コマンドタスクを使用して、ファイルの移動、ファイルのコピー、ファイルの圧縮や解凍、またはクリーンスクリプトや SQL スクリプトの実行をタスクフローの一部として行えます。

コマンドタスクの出力を使用して、タスクフローの後続のタスクを調整できます。

File Watch タスク

File Watch タスクステップを使用して、定義された場所にあるファイルをリッスンし、ファイルイベントを監視します。File Watch タスクステップでは、コネクタソースタイプを使用して既存のファイルリスナーを選択できます。タスクフローの実行を調整するため、ファイルイベントを使用できます。例えば、ファイルが特定の場所に到着するまで待機してから、次のステップでそのファイルを使用するといったことができます。

取り込みタスク

取り込みタスクステップを使用して、ファイル取り込みタスクをタスクフローの調整に活用します。取り込みタスクステップでは、既存のファイル取り込みタスクを選択できます。

ファイルを中間位置に移動した後で、ファイルをターゲットに転送する前にデータ統合操作を実行することをお勧めします。この場合、取り込みタスクステップをデータタスクステップと組み合わせて使用できます。

例えば、取り込みタスクステップを使用してソースの場所から多数のファイルを読み取り、それらを中間位置に書き込むことができます。次に、データタスクステップを使用してファイルに対してデータ統合操作を実行し、別の取り込みタスクステップを使用して更新済みファイルを最終的なターゲットの場所に書き込むことができます。

サブタスクフロー

既存のタスクフローを埋め込んで再利用するには、サブタスクフローステップを使用します。入力フィールドを設定すると、タスクフローの実行時に入力を行えます。フォールト処理を有効にして、タスクフローのエラー理由を特定することもできます。

決定

タスクフローが特定のフィールドの値に基づいて異なるパスを取るようにする場合は、決定ステップを使用します。

並列パス

タスクフローで複数の項目を同時に実行する場合は、並列パスを使用します。例えば、3つのマッピングタスクを同時に実行できます。タスクフローは、並列パスステップのすべての項目を実行し、次のステップに移動します。

ジャンプ

タスクフローのある部分から別の部分にジャンプする場合は、ジャンプステップを使用します。

終了

タスクフローの完了時に使用する必要のある HTTP ステータスコードを定義するには、終了ステップを使用します。

待機

特定の期間、タスクフロー実行を一時停止する場合は、待機ステップを使用します。

スロー

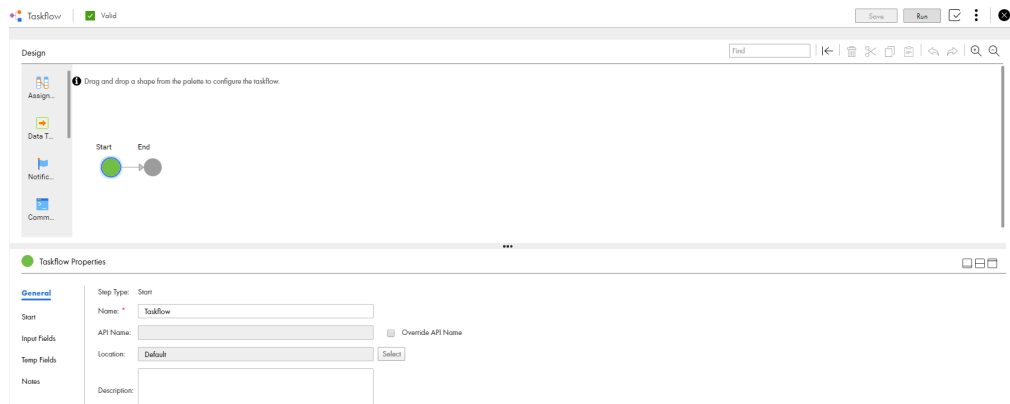
スローステップを使用して、フォールトをキャッチし、フォールトの詳細を返し、タスクフローの後続ステップが実行されないようにします。スローステップは中断ステップです。そのため、フォールトが発生

すると、スローステップはタスクフローの実行を停止し、タスクフローのステータスを「失敗」に設定します。

タスクフローの作成

タスクフローを最初から作成するか、またはタスクフローテンプレートを使用します。

1. Data Integration で、**[新規]** > **[タスクフロー]** > **[タスクフロー]** > **[作成]** をクリックします。



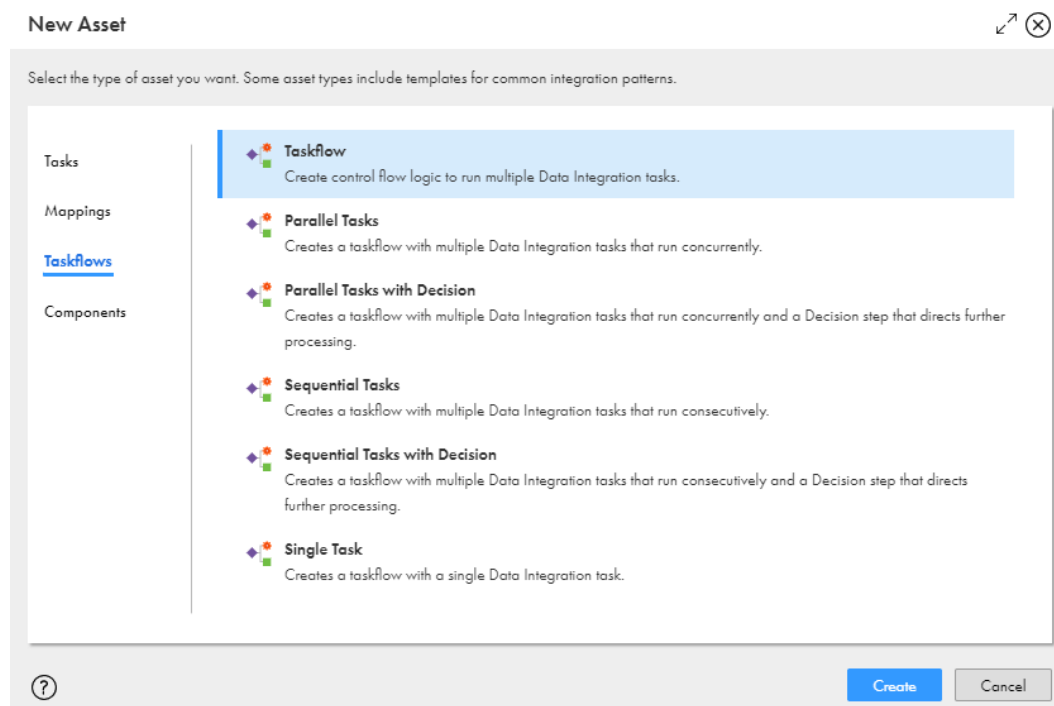
2. タスクフローの一般的なプロパティ、開始プロパティ、入力フィールド、および一時フィールドを設定します。必要に応じて、メモを追加します。
3. タスクフローにステップを追加します。
例えば、データタスクステップを使用して、データ転送タスク、動的マッピングタスク、マッピングタスク、PowerCenter タスク、または同期タスクを追加できます。既存のタスクフローを埋め込んで再利用するには、サブタスクフローステップを使用できます。
4. タスクフローを検証して保存します。無効なタスクフローは実行できません。

タスクフローテンプレート

タスクフローを最初から作成する代わりに、タスクフローテンプレートを使用します。ビジネスニーズに基づいてあらかじめ作成されたテンプレートから選択します。

[新規] > **[タスクフロー]** をクリックして、**[新しいアセット]** ダイアログボックスを開きます。

次の図に、**[新しいアセット]** ダイアログボックスを示します。



【新しいアセット】 ダイアログボックスでタスクフローテンプレートを選択すると、データ統合ページが開き、タスクフローテンプレートのコピーが表示されます。

例えば、複数のタスクを連続して実行する場合は、シーケンシャルタスクテンプレートを使用します。連続したタスクの後で、タスクフローを一時停止してから別のパスを取る必要がある場合は、待機ステップと並列パスステップをテンプレートに追加します。

タスクフローテンプレートを変更するとしても、特定のインスタンスのみが変更されます。テンプレートそのものは変更されません。

基本

開始ステップと終了ステップがある基本キャンバスが必要な場合は、【タスクフロー】ステップを選択します。必要に適したタスクフローを作成および設定できます。

並列タスク

主要な要件が2つ以上のデータ統合タスクを並列に実行することである場合、【並列タスク】テンプレートを選択します。開始ステップ、並列パスステップ、および終了ステップを含むタスクフローで開始します。

キャンバスの任意のポイントで他のステップを追加できます。

ディシジョンがある並列タスク

2つ以上のデータ統合タスクを並列に実行し、いずれかのタスクの結果に基づいて決定を下すことが主要な要件である場合は、【ディシジョンがある並列タスク】テンプレートを選択します。開始ステップ、並列パスステップ、ディシジョンステップ、および終了ステップを含むタスクフローで開始します。

シーケンシャルタスク

主要な要件が2つのデータ統合タスクを順次実行することである場合は、**【シーケンシャルタスク】** テンプレートを選択します。開始ステップ、2つのデータタスクステップ、および終了ステップを含むタスクフローで開始します。

ディシジョンがあるシーケンシャルタスク

2つのデータ統合の連続タスクを実行し、いずれかのタスクの結果に基づいて決定を下すことが主要な要件である場合は、**【ディシジョンがあるシーケンシャルタスク】** テンプレートを選択します。開始ステップ、2つのデータタスクステップ、ディシジョンステップ、および終了ステップを含むタスクフローで開始します。

単一タスク

例えば、主要な要件が1つのデータ統合タスクを日次または週次スケジュールで実行することである場合、**【単一タスク】** テンプレートを選択します。開始ステップ、データタスクステップ、および終了ステップを含むタスクフローで開始します。

タスクフロープロパティの設定

タスクフロープロパティは、タスクフローの名前と場所、タスクフローが使用するフィールド、およびタスクフローを呼び出して実行するために必要な、その他の情報を定義します。

タスクフロープロパティを設定するには、タスクフローを作成し、開始ステップをクリックして**【プロパティ】** セクションにアクセスします。必要に応じて、キャンバスの空の領域をクリックして**【プロパティ】** セクションにアクセスします。

タスクフローの全般プロパティ、タスクフローバインディングとアクセスの詳細、入力フィールド、出力フィールド、一時フィールド、詳細プロパティ、およびメモを定義します。

全般的なプロパティ

タスクフローには、次の一般的なプロパティを指定できます。

プロパティ	説明
名前	必須。タスクフローを識別するためのわかりやすい名前。 異なるフォルダに保存されたタスクフローは、同じ名前を持つことができます。 名前は 80 文字を超えることはできません。
API 名のオーバーライド	オプション。タスクフローをパブリッシュしたときに自動生成される API 名を、指定した名前でオーバーライドします。このオプションを選択すると、 【API 名】 フィールドが使用可能になります。

プロパティ	説明
API 名	<p>【API 名のオーバーライド】 オプションを選択した場合は必須です。</p> <p>タスクフローに対して自動生成された API 名をオーバーライドする一意の API 名です。このフィールドで指定した API 名が、生成されたサービス URL で使用されます。</p> <p>API 名に使用できるのは、英数字、アンダースコア (_)、およびハイフン (-) のみで、80 文字を超える名前は使用できません。</p> <p>パブリッシュされたタスクフローの API 名前を変更するには、まずタスクフローのパブリッシュを解除する必要があります。次に、API 名前を変更し、タスクフローを再パブリッシュします。</p> <p>API 名をオーバーライドしてタスクフローをインポートすると、インポートされたタスクフローでは指定した API 名が使用されます。ただし、指定した名前と同じ API 名を持つ既存のタスクフローがある場合、データ統合はコピーまたはインポートされたタスクフローの自動生成 API 名を使用して、重複を回避します。また、データ統合では、【API 名のオーバーライド】 フィールドも無効になります。同様に、タスクフローをコピーすると、データ統合はコピーされたタスクフローに自動生成される API 名を使用して重複を回避し、【API 名のオーバーライド】 フィールドも無効になります。</p> <p>API 名をオーバーライドしてタスクフローを別の場所に移動した場合、データ統合では、指定した API 名が保持されます。</p>
場所	<p>タスクフローを保存するプロジェクトとフォルダを選択します。【選択】 をクリックして、フォルダに移動します。</p> <p>タスクフローを作成する前に、プロジェクトとフォルダを作成する必要があります。タスクフロー作成ページからプロジェクトまたはフォルダを作成することはできません。</p> <p>【参照】 ページが現在アクティブになっていて、プロジェクトまたはフォルダが選択されている場合、アセットのデフォルトの場所はその選択されているプロジェクトまたはフォルダです。そうでない場合、デフォルトの場所は直近で保存されたアセットの場所です。</p>
説明	タスクフローの説明。

開始プロパティ

【開始】 タブを使用して、タスクフローのバインディングタイプおよびアクセスの詳細を定義します。

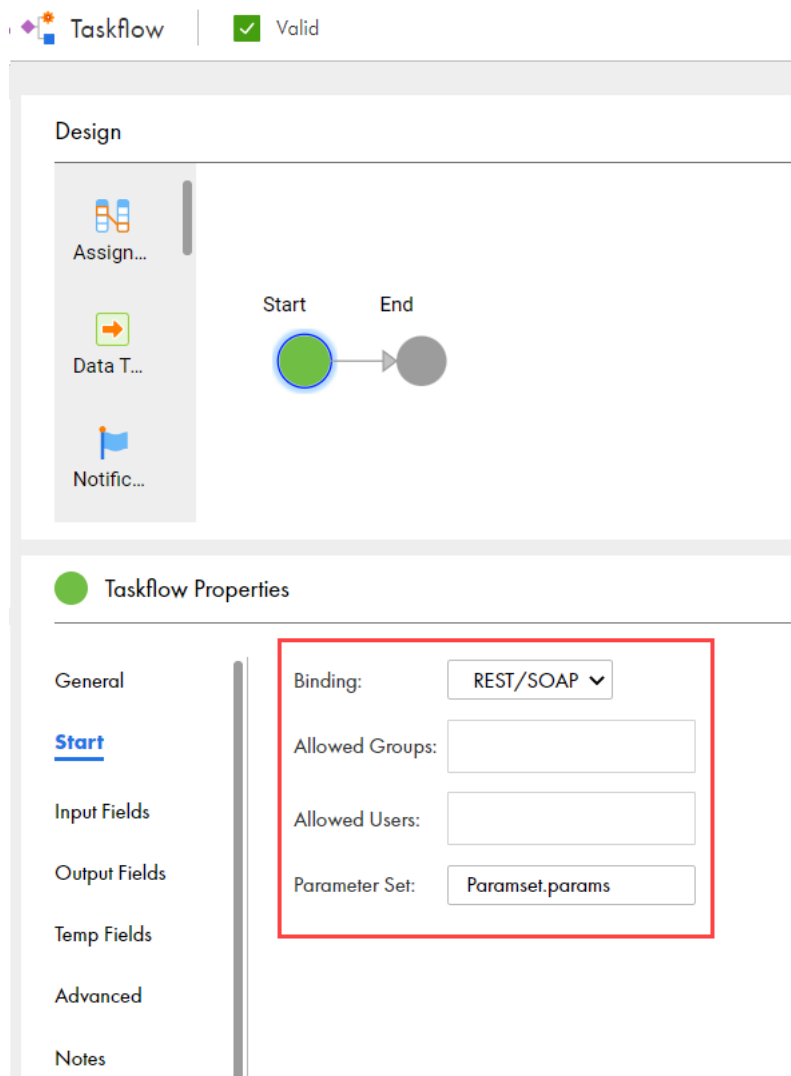
タスクフローバインディング

【バインディング】 プロパティではタスクフローの呼び出しおよび実行方法を定義します。次の値から選択する事ができます。

REST/SOAP

【REST/SOAP】 バインディングタイプを選択すると、REST または SOAP エンドポイントを使用してタスクフローを実行できます。**【許可されたグループ】** および **【許可されたユーザー】** フィールドを使用して、パブリッシュされたタスクフローを API として実行できるグループおよびユーザーを定義できます。

次の図は、**【REST/SOAP】** に設定されたバインディング、**【許可されたグループ】** フィールド、**【許可されたユーザー】** フィールド、および **【パラメータセット】** フィールドを示しています。

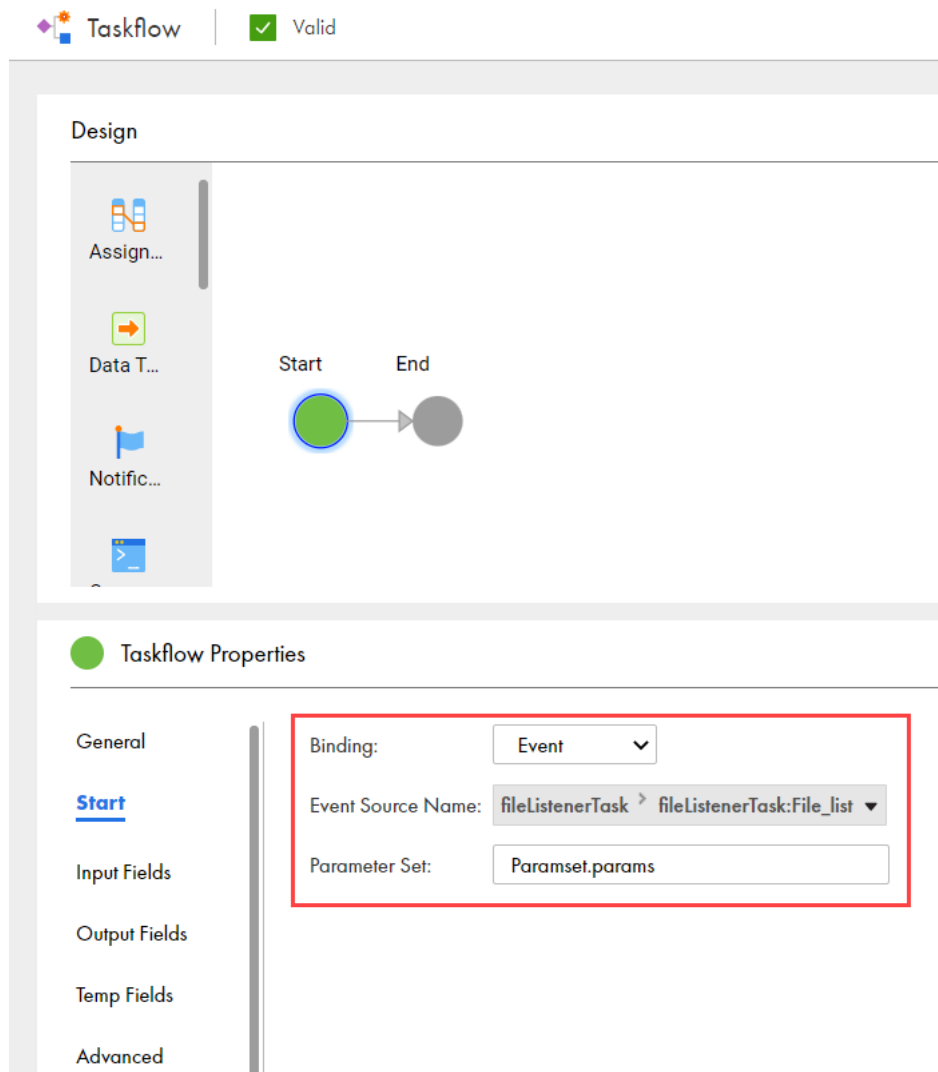


イベント

【イベント】 バインディングタイプを選択すると、指定したイベントが発生したときにタスクフローが呼び出されます。例えば、ファイルシステムのファイルの到着などのイベントの際にタスクフローを呼び出す事ができます。【イベントソース名】 フィールドは、イベント用に作成したファイルリスナを選択できる場合に使用できます。

ファイルリスナの詳細については、『コンポーネント』を参照してください。

次の図は、【イベント】 に設定されたバインディング、【イベントソース名】 フィールド、および【パラメータセット】 フィールドを示しています。



タスクフローをパブリッシュした後は、バインディングの詳細を編集する事は出来ません。バインディングの詳細を編集するには、タスクフローをパブリッシュ解除する必要があります。

タスクフローのアクセス

タスクフローが **[REST/SOAP]** バインディングタイプを使用する場合、**[許可されたグループ]** および **[許可されたユーザー]** フィールドに、パブリッシュされたタスクフローを API として実行できるグループおよびユーザーを定義できます。これらのうち、いずれかのカテゴリに当てはまるユーザーは、実行時にタスクフローサービス URL にアクセスできます。

[許可されたグループ] フィールドと **[許可されたユーザー]** フィールドをどちらも設定しない場合、データ統合はタスクフローサービス URL を生成しません。タスクフローを実行し、スケジュール設定できます。しかし、タスクフローを API として実行する事は出来ません。

注: **[許可されたグループ]** および **[許可されたユーザー]** フィールドでは、API として実行中のタスクフローのアクセスの詳細を定義します。タスクフローデザイナーからのタスクフローの実行またはファイルリサナ経由のタスクフローの呼び出しの権限を設定するには、Administrator でロールを作成し、必要な権限を割り当てる必要があります。

以下のプロパティを設定します。

許可されたグループ

実行時にタスクフローサービス URL にアクセスできるグループを定義します。

ユーザーグループにタスクフローサービス URL へのアクセス権を付与する場合は、**【許可されたグループ:】** オプションを使用します。例えば、「Order Approvers」というグループがある場合を考えます。**【許可されたグループ:】** フィールドに「Order Approvers」と入力すると、このグループ内のすべてのユーザーがタスクフローサービス URL へのアクセス権を持ちます。

次の図は、**【許可されたグループ:】** フィールドに Order Approvers グループが追加されているところを示します。

Allowed Groups:

Order Approvers ✕

Allowed Users:

【許可されたグループ:】 フィールドには、複数のグループを指定できます。

許可されたユーザー

実行時にタスクフローサービス URL にアクセスできるユーザーを定義します。

特定のユーザーにタスクフローサービス URL へのアクセス権を付与する場合は、**【許可されたユーザー:】** フィールドを使用します。次の図では、**【許可されたユーザー:】** フィールドにユーザー jsmith が表示され、**【許可されたグループ:】** に Order Approvers が表示されています。

Allowed Groups:

Order Approvers ✕

Allowed Users:

jsmith ✕

Order Approvers グループのユーザーと、ユーザー jsmith は、タスクフローサービス URL へのアクセス権を持ちます。

【許可されたユーザー:】 フィールドには、複数のユーザーを指定できます。

パラメータセット

タスクフローの入力パラメータの値を指定するパラメータセットを定義します。

パラメータセットにはタスクフローレベルのセクションとパラメータが含まれており、これは、ParamSetCli ユーティリティを使用して、Informatica が管理するクラウドホステッドリポジトリにアップロードされます。

パラメータセットの詳細については、[「パラメータセット」 \(ページ 57\)](#)を参照してください。

入力フィールド

【入力フィールド】 セクションを使用して、ステップの先頭でタスクフローが使用するフィールドを追加します。

タスクフローを実行するときに渡す入力フィールドを定義できます。次のタイプの入力フィールドを作成できます。

- **単純型。** チェックボックス、日付、日付/時刻、時間、数値、整数、テキストなどの一般的なデータ型を使用する単純型フィールドを作成します。
- **カスタムタイプ。** タスクフローに追加されたオブジェクトを使用するためのカスタムタイプフィールドを作成します。

作成する入力フィールドごとに次のプロパティを入力します。

プロパティ	説明
名前	入力フィールドの名前。
タイプ	入力フィールドのタイプ。例えば、[チェックボックス]、[日付]、[日付/時刻]、[時間]、[番号]、[整数]、または[テキスト]を選択します。 また、単純型またはカスタム型の入力フィールドを追加することもできます。入力フィールドを追加するには、[タイプをさらに表示]を選択し、[タイプの編集] ダイアログボックスで、[カテゴリ] に [単純タイプ] または [カスタムタイプ] を選択します。
説明	入力フィールドの説明。
必須	タスクフローを実行するために入力フィールドが必要かどうかを示します。

ファイルリснаにより呼び出されるタスクフローの入力フィールド

タスクフローのバインディングタイプに [イベント] を選択し、ファイルリснаを選択する場合、データ統合は、到着してリснаイベントの一部として更新または削除されたフィールドの詳細を格納する入力フィールドを作成します。入力フィールドはファイルリснаの名前を取得します。この入力フィールドを編集または削除する事は出来ません。ファイルリснаにより呼び出されたタスクフローの入力フィールドを追加する事も出来ません。

タスクフローが完了した後で、[マイジョブ] ページにアクセスし、タスクフローインスタンスをクリックし、[入力フィールド] タブをクリックして、到着してファイルリснаイベントの一部として更新または削除されたファイルの詳細を表示できます。入力フィールドには、到着して更新または削除された各ファイルについて、次の情報が表示されます。

- ファイルが使用可能なパス
- ファイルの名前
- ファイルのサイズ (バイト)
- 最終変更時刻 (ミリ秒)

ファイルリснаの実行や、ファイルリснаの各実行ジョブで発生するイベントを監視する事ができます。Monitor の [ファイル転送ログ] ページには、ファイルリснаのログエントリが一覧表示されます。

注: タスクフローのバインディングタイプに [REST/SOAP] を選択してから、それを [イベント] に変更した場合、データ統合は以前に追加した入力フィールドを削除します。

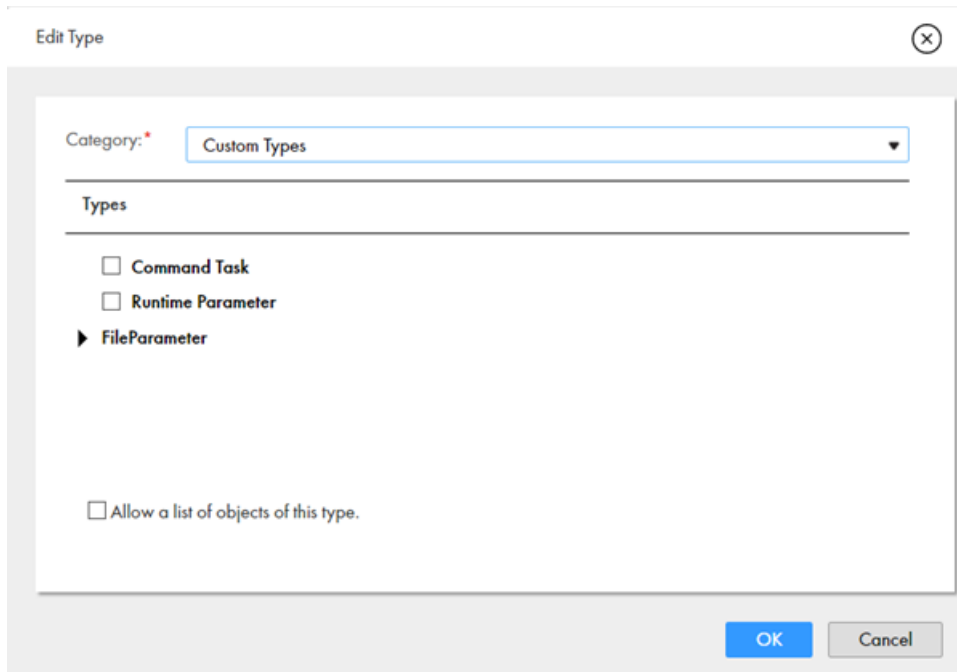
カスタムタイプを使用した入力フィールドの作成

タスクフローのタスクで使用可能なオブジェクトを使用して、カスタムタイプの入力フィールドを作成できます。カスタムタイプの入力フィールドを作成するには、タスクフローに追加されたデータ統合タスクを含む 1 つ以上のステップが必要です。

1. [参照] ページで、カスタムタイプの入力フィールドを作成するタスクフローに移動します。
2. タスクフローにステップを追加します。例えば、データタスクステップと取り込みタスクステップの追加などを行います。
3. タスクフローの [開始] ステップをクリックします。
4. [入力フィールド] タブをクリックします。
5. [追加] アイコンをクリックします。
6. [タイプ] カラムで、[タイプをさらに表示] を選択します。

【**タイプの編集**】 ダイアログボックスが表示されます。

次の図に、【**タイプの編集**】 ダイアログボックスを示します。



7. 【**カテゴリ**】 リストで【**カスタムタイプ**】を選択します。
タスクフローで使用可能なオブジェクトのリストが【**タイプ**】セクションに表示されます。
8. 入力として渡すオブジェクトを選択し、【**OK**】をクリックします。

出力フィールド

【**出力フィールド**】 セクションを使用して、出力フィールドをタスクフローに追加します。

親タスクフローの後続のステップで、サブタスクフローの出力フィールドを変数として使用できます。ただし、サブタスクフローが失敗した場合、サブタスクフローの出力フィールド値は親タスクフローに送信されません。親タスクフローを実行すると、サブタスクフローステップの出力フィールドを、データ統合の【**マイジョブ**】ページ、および Monitor の【**すべてのジョブ**】ページと【**実行中のジョブ**】ページで確認できます。

次のタイプの出力フィールドを作成できます。

- **単純型**。チェックボックス、日付、日付/時刻、時間、数値、整数、テキストなどの一般的なデータ型を使用する単純型フィールドを作成します。
- **カスタムタイプ**。タスクフローに追加されたオブジェクトを使用するためのカスタムタイプフィールドを作成します。

作成する出力フィールドごとに次のプロパティを入力します。

プロパティ	説明
名前	出力フィールドの名前です。
タイプ	出力フィールドのタイプです。例えば、[チェックボックス]、[日付]、[日付/時刻]、[時間]、[番号]、[整数]、または[テキスト]を選択します。 また、単純型またはカスタムタイプの入力フィールドを追加することもできます。入力フィールドを追加するには、[タイプをさらに表示]を選択し、[タイプの編集]ダイアログボックスで、[カテゴリ]に[単純タイプ]または[カスタムタイプ]を選択します。
説明	出力フィールドの説明です。
初期値	出力フィールドの初期値です。

一時フィールド

タスクフローステップで使用する一時フィールドを作成します。タスクフローは、一時フィールドを内部的に使用します。一時フィールドは、タスクフローの入力または出力には表示されません。

例えば、タスクフローに決定ステップがある場合には、一時フィールドを定義できます。

一時フィールドごとに次のプロパティを定義します。

プロパティ	説明
名前	一時フィールドの名前。
タイプ	一時フィールドのタイプ。例えば、[チェックボックス]、[日付]、[日付/時刻]、[時間]、[番号]、[整数]、または[テキスト]を選択します。 また、単純型またはカスタム型の一時フィールドを追加することもできます。入力フィールドを追加するには、[タイプをさらに表示]を選択し、[タイプの編集]ダイアログボックスで、[カテゴリ]に[単純タイプ]または[カスタムタイプ]を選択します。
説明	一時フィールドの説明。
初期値	一時フィールドの初期値です。この値は、タスクフローの実行中に変更される可能性があります。

一部の一時フィールドは、特に追加しなくても表示されます。タスクフローにタスクを追加すると、対応する一時フィールドが表示されます。一時フィールドの【名前】は、データタスクステップの名前です。一時フィールドの【タイプ】は、データタスクステップに追加するタスクの名前です。【説明】を入力し、必要に応じて【必須】を選択します。

タスクフローにデータタスクステップを含めると、開始ステップの【一時フィールド】タブに[データタスク]フィールドが表示されます。[データタスク]フィールドは、タスクの入力パラメータを表します。

詳細プロパティ

同時実行を無効にするようにタスクフローを設定できます。また、タスクフローレベルで障害が発生したら一時停止するようにタスクフローを設定し、一時停止時には指定した受信者に電子メール通知を送信することもできます。

タスクフローには、次の詳細プロパティを定義できます。

プロパティ	説明
同時実行の無効化	タスクフローの同時実行を無効にします。 このオプションを選択すると、タスクフローは、別のインスタンスがまだ実行中の場合に新しいインスタンスを作成しません。 前のインスタンスが完了する前にタスクフローを実行すると、要求は失敗します。 注: このオプションは、サブタスクフローステップでタスクフローを使用する場合は適用されません。
フォールト時に一時停止	タスクフローレベルでフォールトが発生した場合、タスクフローを一時停止します。 【フォールト時に一時停止】 プロパティは、データタスクステップで定義する次のプロパティよりも優先されます。 - 【タスクフローを完了時に終了 - このタスクが失敗した場合】 - 【タスクフローを完了時に終了 - このタスクが実行されない場合】
一時停止時に電子メールを送信	フォールトの発生でタスクフローが一時停止すると、電子メール通知を送信します。 【一時停止時に電子メールを送信】 オプションを選択すると、【電子メールの宛先】、【電子メールの CC】、【電子メールの件名】、【電子メールの本文】 の各フィールドが使用可能になります。
電子メールの宛先	電子メール通知の主な受信者を定義します。有効な受信者の電子メールアドレスを 1 つ以上入力します。カンマ (,) またはセミコロン (;) を使用して電子メールアドレスを区切ります。 一時停止時に電子メール通知を送信するようにタスクフローを設定した場合は必須です。
電子メールの CC	電子メール通知のコピーを受信する必要がある受信者を定義します。有効な受信者の電子メールアドレスを 1 つ以上入力します。カンマ (,) またはセミコロン (;) を使用して電子メールアドレスを区切ります。
電子メールの件名	電子メールについて紹介する、短く説明的な件名を入力します。
電子メールの本文	電子メールで送信するコンテンツを定義します。 【コンテンツの編集】 をクリックしてリッチテキストエディタを開き、太字、斜体、下線、箇条書き、インデント、フォントなどの書式設定オプションを使用します。表やリンクも挿入できます。

同時実行の無効化のシナリオ

タスクフローについて【同時実行の無効化】オプションを有効にする場合は、次のシナリオを考慮してください。

- 前のタスクフローインスタンスが一時停止状態の場合、タスクフローを再度実行しても、データ統合はタスクフローインスタンスを実行しません。
- タスクフローインスタンスが実行中の場合は、データ統合の【マイジョブ】ページにある【ここから実行します】オプション、および Monitor の【すべてのジョブ】と【実行中のジョブ】ページを使用して、正常に実行されたステップから以前のタスクフローインスタンスを実行できます。
- 親タスクフローとサブタスクフローは独立して実行されます。例えば、サブタスクフローを含むタスクフローがあり、どちらのタスクフローでも【同時実行の無効化】オプションが有効になっているとします。親タ

タスクフローを実行すると、子タスクフローがサブタスクとして実行されます。タスクフローまたはサブタスクフローを独立して実行すると、初めて問題なく実行されます。ただし、前のインスタンスがまだ実行されている間に同じタスクフローまたはサブタスクフローを再度実行すると、エラーが発生します。

- エンドポイントを使用してタスクフローを実行するときに、タスクフローの前のインスタンスがまだ実行されていると、エラーが発生します。ただし、同じタスクフローをタスクフローデザイナーから、または API として何度か実行すると、成功メッセージが表示されます。これは、タスクフローを実行すると、データ統合が要求を受け取り、すぐに 200 OK という応答を送信するためです。データ統合は、タスクフロー実行要求を後で処理します。これはユーザーエクスペリエンスの向上になります。タスクフローは、その複雑さによっては実行に時間がかかる場合があります。

- **【次を使用して実行】** オプションをクリックしてタスクフローを実行するとき、および **【すべて実行】** をクリックして複数の入力を使用してタスクフローを実行するとき、一貫性のない動作が見られる場合があります。

例えば、タスクフローの入力が 3 つあり、3 つのすべての入力を使用してタスクフローを実行するとします。これらの 3 つの要求はすべて一度にトリガされます。したがって、タスクフローは、1 つの入力で実行される場合や、2 つの入力で実行される場合があります。

これは、複数のタスクフロー入力を伴うタスクフローを実行する場合、複数の要求が一度にトリガされるためです。データ統合は、実行中の既存のインスタンスがあるかどうかを確認します。各要求は最初の要求と見なされ、タスクフローの実行が試行されます。

メモ

【メモ】 フィールドを使用して、ユーザーまたは他のユーザーが必要とする可能性がある情報を追加します。ここに入力したメモは、データ統合ページに表示されます。タスクフローを実行すると、これらのメモは表示されません。

例えば、タスクフローを実行する前に完了する必要があるタスクに関するアラームを追加するには、**【メモ】** フィールドを使用します。

タスクフローステップのプロパティの設定

タスクフローにステップを追加する場合は、各ステップに関連付けられているプロパティを設定します。

割り当てステップ

割り当てステップを追加すると、次のプロパティを設定できます。

名前

完全修飾フィールド名。

フィールド名は、**【開始】** > **【フィールド】** で定義されているフィールドの一覧から選択します。

割り当て

フィールドの値を取得するソース。表示されるフィールドは、**【開始】** > **【プロパティ】** > **【入力フィールド】** または **【一時フィールド】** に対して定義されているデータ型に応じて異なります。

例えば、**【入力フィールド】** のデータ型を **日付/時刻** と定義します割り当てステップのフィールドには、次の **【値】** オプションが表示されます。

- 特定の日付

- 現在からの時間
- 本日からの日数
- 前/後の日数
- フィールド
- 計算式

割り当てステップでは、ランタイムパラメータを上書きできます。

ランタイムパラメータの詳細については、[Runtime Parameters in Taskflows \(ページ 53\)](#)を参照してください。

データタスクステップ

データタスクステップを追加するときに、いくつかのプロパティを設定します。

次のセクションでは、データタスクステップのプロパティについて説明します。

全般

全般プロパティで、データタスクステップにわかりやすい名前を指定できます。

データタスク

データタスクステップのプロパティで、タスクフローに追加するタスクを、既存のタスクリストから選択します。

注: タスクフローに追加する既存のタスクが必要です。タスクフロー作成プロセスでタスクを作成することはできません。

マッピングタスクをデータタスクステップに追加すると、説明、入力フィールド、および出力フィールドが表示されます。入力フィールドには、マッピングタスクが使用する入出力パラメータが表示されます。出力フィールドには、タスクフローの実行後にマッピングタスクが返す出力フィールドが表示されます。

データタスクステップで **[追加]** アイコンをクリックすると、次のいずれかのビューが表示されます。

- データタスクステップにタスクが含まれている場合、タスクの編集不可能なビューが開きます。
- データタスクステップにタスクが含まれていない場合は、タスクを選択できるダイアログボックスが表示されます。

タスクをデータタスクステップに追加すると、テキスト型の対応するタスクフロー一時フィールドが作成されます。データタスクステップにタスクを追加すると、一時フィールドのタイプはタスクの名前になります。詳細については、「[一時フィールド \(ページ 20\)](#)」を参照してください。

入力フィールド

タスクをタスクフローに追加すると、**[入力フィールド]** セクションが表示されます。

上書きできるパラメータがタスクに含まれている場合は、入力フィールドを追加できます。入力フィールドのプロパティを設定して、データ統合ランタイムパラメータを上書きできます。ランタイムパラメータの詳細については、「[データタスクステップでのパラメータまたはパラメータファイルの上書き \(ページ 54\)](#)」を参照してください。

データタスクステップで PowerCenter タスクまたはマッピングタスクを使用する場合は、タスクの入力パラメータと入出力パラメータの値を上書きできます。

データタスクステップでマッピングタスクを使用する場合は、マッピングタスクのパラメータファイルディレクトリとパラメータファイル名を上書きできます。データタスクステップでルックアップトランスフォーマーションを含むマッピングタスクを使用する場合、ルックアップオブジェクトとルックアップ条件の値を上書きできます。

データタスクステップで動的マッピングタスクを使用する場合、**【ジョブフィルタ】** という名前の入力パラメータを追加できます。入力フィールドの名前は編集できません。ただし、タスクフローで実行する動的マッピングタスクからグループとジョブを指定することはできます。

グループとジョブを指定するには、**【編集】** をクリックしてから、<group_name>.<job_name>という値をコンテンツタイプとともに入力フィールドに入力します。例えば、動的マッピングタスクから Group_1 と Job_1 を実行する場合は、**【ジョブフィルタ】** 入力フィールドに Group_1.Job_1 という値を入力します。

【ジョブフィルタ】 入力フィールドを追加しない場合、タスクフローはデフォルトで、動的マッピングタスクで使用可能なすべてのジョブを指定した順序で実行します。

出力フィールド

タスクフローに同期タスクまたは PowerCenter タスクを追加すると、**【出力フィールド】** セクションが表示されます。**【出力フィールド】** セクションは、タスクの実行時に表示される出力データフィールドの完全な一覧です。

次の図は、表示される出力フィールドを示しています。

Output Fields:

Name	Type
Run Id	Integer
Task Status	Text
Success Source Rows	Integer
Failed Source Rows	Integer
Success Target Rows	Integer
Failed Target Rows	Integer
Start Time	Date Time
End Time	Date Time
Error Message	Text

マッピングタスクが詳細クラスタで実行される場合は、次のフィールドが表示されます。

Output Fields

Name	Type
Run Id	Integer
Log Id	Text
Task Status	Text
Success Target Rows	Integer
Failed Target Rows	Integer
Start Time	Date Time
End Time	Date Time
Error Message	Text

注: 詳細クラスタで実行されるマッピングタスクの場合、タスクの実行時に、成功したソース行と失敗したソース行は取り込まれません。

データ転送タスクまたは動的マッピングタスクを使用する場合、次のフィールドが表示されます。

Output Fields

Name	Type
Run ID	Int
Status	String
Success Rows	Int
Failed Rows	Int
Error Message	String

各出力フィールドの値を表示するには、タスクフローを実行し、**【タスクフローインスタンスの詳細】** ページに移動します。**【タスクフローインスタンスの詳細】** ページの詳細については、Monitor のヘルプを参照してください。

データ決定または割り当てのステップで出力フィールドを使用できます。

例えば、値式の一時フィールドを作成し、次の式を使用してデータをフィールドに割り当てます。

```
if(
($temp.DataTask1[1]/output[1]/Failed_Target_Rows < 0 or
$temp.DataTask1[1]/output[1]/Task_Status = '1')
and
($temp.DataTask2[1]/output[1]/Success_Target_Rows > 0
and $temp.DataTask2[1]/output[1]/Failed_Target_Rows = 0)
and $temp.DataTask3[1]/output[1]/Success_Target_Rows > 0)
then 'Pass'
else 'Fail'
```

決定ステップで一時フィールドを使用すると、次の条件が満たされた場合、タスクフローは Pass パスを受け取ります。

- データタスク 1 には、失敗したターゲット行はありません。またはデータタスク 1 は、正常に実行されています。
- データタスク 2 には、少なくとも 1 つの成功したターゲット行があります。
- データタスク 2 では、失敗したターゲット行はゼロです。
- データタスク 3 には、少なくとも 1 つの成功したターゲット行があります。

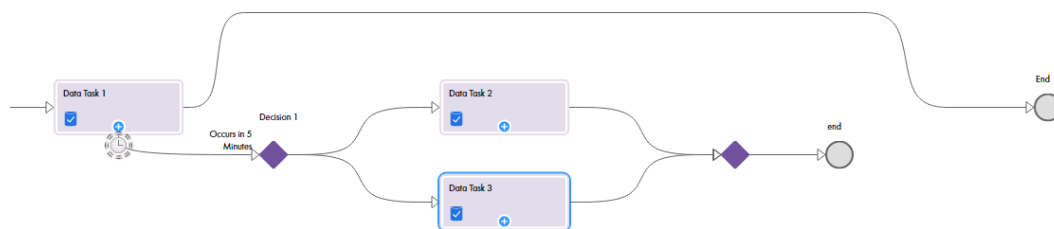
タイマイイベント

タスクにタイマを追加するには、次の**イベント**プロパティを入力します。

タイマイイベントを使用して、スケジュールに基づいてアクションを実行します。アクションは、特定の時刻または間隔の後のいずれかになります。

データタスクステップにタイマを追加すると、新しい分岐が表示されます。この分岐にイベントを追加し、特定の**【時間】** または間隔**【以降】** に実行するかどうかを指定します。

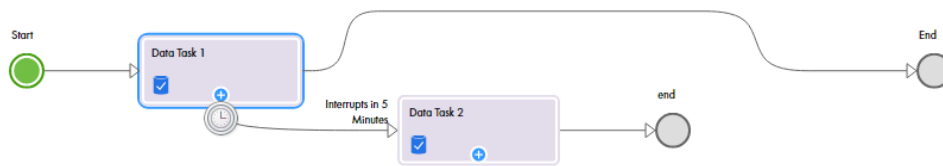
次の図では、タイマ分岐のイベント（データ決定ステップ）が、メインデータタスクの 5 分後に発生します。



タイマが起動すると、タスクフローは常にタイマ分岐全体を通して実行されます。データタスク 1 が決定 1 の前に終了する場合、タイマ分岐は実行されません。

タイマでメインデータタスクを中断する場合は、**【中断】** を選択します。中断タイマを設定すると、メインデータタスクが中断され、タスクフローはタイマ設定に基づいてイベントのみを実行します。

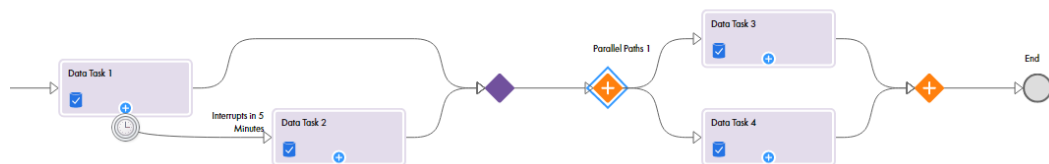
次の図は、メインデータタスクが開始されてから 5 分後に発生する中断タイマ設定を示しています。



タイマ分岐に基づくイベントの場合、データタスク 2 が実行され、データタスク 1 が中断されます。タスクフローはタイマ分岐に従います。つまり、タスクフローはデータタスク 2 を実行してから終了します。

中断タイマのタイマ分岐の終了ステップを削除すると、タイマ分岐はメイン分岐に再び参加します。

次の図は、終了ステップが削除された中断タイマ分岐を示しています。



タイマイイベント（データタスク 2）は、5 分後に実行され、データタスク 1 を中断します。タイマ分岐は、メイン分岐に再び参加します。タスクフローは、データタスク 2、並列パスステップを実行し、終了します。

中断タイマを使用する場合、メインデータタスクには、このタスクフローインスタンスに対する出力はありません。タスクフローのジョブの詳細ページには、メインデータタスクの出力フィールドが表示されません。

データタスクステップがタイマ、中断、または非中断の前に完了した場合、そのデータタスクステップに対してタイマは起動しません。

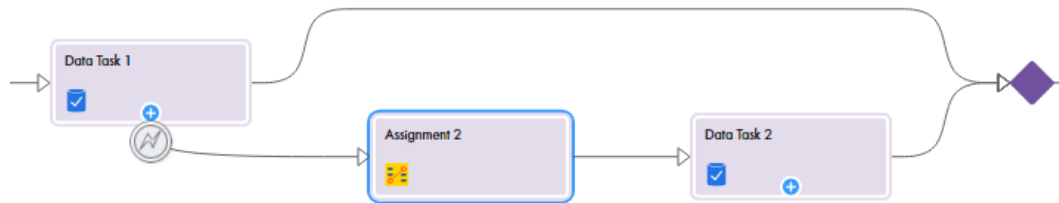
エラー処理

[エラー処理] セクションを使用すると、データタスクステップが警告またはエラーを検出したときにタスクフローがどのように動作するかを指定できます。データタスクステップに関連付けられたタスクが失敗した場合、または実行されなかった場合のタスクフローの動作を設定することもできます。

タスクを選択した後に、次のエラー処理のプロパティを入力します。

プロパティ	説明
警告時	<p>データタスクステップで警告が発生したときにタスクフローが受け取るパス。</p> <p>警告は、データタスクステップが誤った仕方で完了したか、または完了しなかった場合に発生します。例えば、データタスクステップで、テーブル A からテーブル B に 25 行のうち 20 行のみをコピーする場合、警告が表示されます。</p> <p>次のオプションから選択できます。</p> <ul style="list-style-type: none"> - 警告を無視して次のステップに進むには、【無視】を選択します。 <p>注: 通知タスクステップが後に続くデータタスクステップに 【無視】を選択してデータタスクが失敗した場合は、受信する電子メール通知にはフォールトの詳細は含まれません。電子メールでフォールトの詳細を確認するには、【カスタムエラー処理】を選択します。</p> <ul style="list-style-type: none"> - 警告が発生したときにタスクフローを中断状態に移行させるには、【タスクフローの中断】を選択します。【すべてのジョブ】、【実行中のジョブ】、および【マイジョブ】ページからタスクフローインスタンスを再開できます。 <p>タスクフローは、中断されたステップから再開します。警告の理由がわかっている場合は、問題を修正してからタスクフローを再開してください。</p> <p>デフォルト値: 無視</p>
エラー発生時	<p>データタスクステップでエラーが発生したときにタスクフローが受け取るパス。</p> <p>データタスクステップが失敗するとエラーが発生します。例えば、データタスクがテーブル A をテーブル B にコピーしない場合、エラーが表示されます。</p> <p>次のオプションから選択できます。</p> <ul style="list-style-type: none"> - 【無視】を選択してエラーを無視し、次の手順に進みます。 - エラーが発生したときにタスクフローを中断状態に移行させるには、【タスクフローの中断】を選択します。【すべてのジョブ】、【実行中のジョブ】、および【マイジョブ】ページからタスクフローインスタンスを再開できます。 <p>タスクフローは、中断されたステップから再開します。エラーの理由がわかっている場合は、問題を修正してからタスクフローを再開してください。</p> <ul style="list-style-type: none"> - 選択した方法でエラーを処理するには、【カスタムエラー処理】を選択します。【カスタムエラー処理】を選択すると、2 つの分岐が表示されます。最初の分岐は、エラーが発生しない場合にタスクフローが続くパスです。2 番目の分岐は、エラーが発生した場合にタスクフローが続くカスタムパスです。 <p>デフォルト値: タスクフローの中断</p>
タスクフローを完了時に終了	<p>データタスクステップに関連付けられたタスクが失敗した場合、または実行されなかった場合のタスクフローの動作。</p> <p>データタスクステップに関連付けられたタスクが失敗した場合、または実行されなかった場合に、完了時に失敗するようにタスクフローを設定できます。タスクが失敗するか実行されなかった場合、タスクフローは後続のステップの実行を続行します。ただし、タスクフローが完了すると、タスクフローのステータスは失敗に設定されます。</p> <p>注: 【フォールト時に一時停止】 タスクフローの詳細プロパティと 【タスクフローを完了時に終了】 プロパティの両方を設定した場合は、【フォールト時に一時停止】 プロパティが優先されます。この場合、データタスクステップに関連付けられたタスクが失敗するか実行されない場合、タスクフローは一時停止されます。タスクフローは、データタスクステップの後の後続のステップを実行しません。</p>

次の図は、割り当てステップと別のデータタスクステップがある、**カスタムエラー処理**パスを示しています。



通知タスクステップ

通知タスクステップは、指定した受信者に通知を送信します。

電子メール通知を送信する通知タスクステップを設定できます。例えば、タスクフローのデータタスクステップで見つかった成功した行とエラー行の数について受信者に知らせる電子メール通知を送信できます。

通知タスクステップのプロパティを定義して、電子メール受信者と電子メールコンテンツを設定できます。次の節では、通知タスクステップのプロパティについて説明します。

全般プロパティ

全般プロパティで、通知タスクステップにわかりやすい名前を指定できます。

通知タスクプロパティ

次の通知タスクステップのプロパティを設定できます。

通知メソッド

送信する通知のタイプ。

このフィールドの値は、デフォルトで【電子メール】に設定されます。この値は編集できません。他の値は将来使用するために予約されています。

電子メールの宛先

必須。電子メール通知のプライマリ受信者。

次のいずれかのオプションを使用して、このフィールドの値を指定します。

- **コンテンツ**。有効な受信者の電子メールアドレスを 1 つ以上入力します。カンマ (,) またはセミコロン (;) を使用して電子メールアドレスを区切ります。
- **フィールド**。このステップを実行するときに、タスクフローデザイナーがこのフィールドに電子メールアドレスを書き込むために使用するフィールドを選択します。タスクフローの他のステップで追加された入力フィールドや一時フィールドを選択できます。
- **式**。式エディタを開いて、このフィールドの値を計算する式を指定します。

デフォルトは【コンテンツ】です。

電子メールの CC

電子メール通知のコピーが送信される必要のある受信者。

次のいずれかのオプションを使用して、このフィールドの値を指定します。

- **コンテンツ**。有効な受信者の電子メールアドレスを 1 つ以上入力します。カンマ (,) またはセミコロン (;) を使用して電子メールアドレスを区切ります。

- **フィールド**。このステップを実行するときに、タスクフローデザイナーがこのフィールドに電子メールアドレスを書き込むために使用するフィールドを選択します。タスクフローの他のステップで追加された入力フィールドや一時フィールドを選択できます。
- **式**。式エディタを開いて、このフィールドの値を計算する式を指定します。

デフォルトは **【コンテンツ】** です。

電子メールの件名

電子メールについて紹介する、短く説明的な件名。

次のいずれかのオプションを使用して、このフィールドの値を指定します。

- **コンテンツ**。電子メールの件名を入力します。
- **フィールド**。このステップを実行するときに、タスクフローデザイナーがこのフィールドに電子メールの件名を書き込むために使用するフィールドを選択します。タスクフローの他のステップで追加された入力フィールドや一時フィールドを選択できます。
- **式**。式エディタを開いて、このフィールドの値を計算する式を指定します。

デフォルトは **【コンテンツ】** です。

電子メールのコンテンツタイプ

電子メールコンテンツで使用する形式のタイプ。

次のいずれかの値を選択します。

- **HTML**。太字、斜体、下線、リスト、インデント、およびフォントなどの形式オプションを使用するには、**【HTML】** を選択します。表やリンクも挿入できます。
- **プレーンテキスト**。フォーマットや特別なレイアウトオプションを指定せずに、通常のテキストを追加するには、**【プレーンテキスト】** を選択します。

デフォルトは **【プレーンテキスト】** です。

電子メールの本文

電子メールで送信するコンテンツ。

次のいずれかのオプションを使用して、このフィールドの値を指定します。

- **コンテンツ**。電子メール本文のコンテンツを入力します。
- **フィールド**。このステップを実行するときに、タスクフローデザイナーがこのフィールドに電子メール本文を書き込むために使用するフィールドを選択します。タスクフローの他のステップで追加された入力フィールドや一時フィールドを選択できます。
- **式**。式エディタを開いて、このフィールドの値を計算する式を指定します。

デフォルトは **【コンテンツ】** です。

電子メールのコンテンツタイプを **【HTML】** として選択した場合は、**【コンテンツの編集】** をクリックするとコンテンツの形式を設定するためのリッチテキストエディタを表示できます。

電子メール通知の例

タスクフロー内のデータタスクの数に基づいて電子メール本文のコンテンツを定義し、HTML コンテンツを含む電子メール通知を送信できます。

単一のデータタスク

単一のデータタスクに対して通知タスクステップを使用する場合、ソースタイプをコンテンツとして電子メール本文にデータタスクの詳細を入力できます。

例えば、次の XQuery 式を渡すことができます。

```
Hi { $user },
  Data task MyDataTask1 with Run Id { $temp.DataTask1/Output/RunId } started at { $temp.DataTask1/Output/
Start_Time } and completed at { $temp.DataTask1/Output/End_Time } with a status of
{ sff:getTaskStatus($temp.DataTask1/Output/Status) }
```

複数のデータタスク

通知タスクステップを使用して複数のデータタスクのサマリーを電子メールで送信する場合、ソースタイプを数式として XQuery 式を使用できます。

例えば、<DataTask1>と<DataTask2>という名前の 2 つのデータタスクを含むタスクフローに対して、次の XQuery 式を渡すことができます。

```
<html>
  <head>Taskflow Tasks Status Summary</head>
  <body>
    <table>
      <tr>
        <td>Task Name</td>
        <td>Job Id</td>
        <td>Status</td>
        <td>Start Time</td>
        <td>End Time</td>
      </tr>
      {
        let $dataTasks := ($temp.DataTask1, $temp.DataTask2)
        for $dataTask in $dataTasks
        return
          <tr>
            <td>{ local-name($dataTask) }</td>
            <td>{ $dataTask/Output/RunId }</td>
            <td>{ sff:getTaskStatus($dataTask/Output/Status) }</td>
            <td>{ $dataTask/Output/StartTime }</td>
            <td>{ $dataTask/Output/EndTime }</td>
          </tr>
      }
    </table>
  </body>
</html>
```

電子メール通知を送信する前に、受信者の場所に基づいてタイムゾーンを変換することをお勧めします。これを行うために、電子メールの件名または電子メールの本文で `infa:format` XQuery 関数を使用することができます。数式を使用してタイムゾーンを変換する方法の詳細については、次のコミュニティ記事を参照してください。

<https://knowledge.informatica.com/s/article/DOC-19342>

通知タスクステップのルールおよびガイドライン

通知タスクステップで **【コンテンツ】** オプションを使用して電子メール本文を指定し、**【HTML】** オプションを使用して電子メールのコンテンツタイプを指定した場合、特定の制限を受けます。

次のガイドラインを考慮します。

- HTML コンテンツを使用すると、有効な形式の電子メールを受信できない場合があります。変数を使用して HTML コンテンツを定義し、式エディタで変数をシリアル化する必要があります。

例として、HTML コンテンツが次のような場合を考えます。

```
<html>
  Order { $output.OrderID } has been submitted for { $input.CustomerEmail }.
  <br/>
  <b>Order details for your records.</b>
  <br/><br/>
  Item Cost: { $temp.InventoryDetails[1]/ItemCostPrice }
  Item Count: { $temp.InventoryDetails[1]/ItemCount }
```

```

Item Sell Price: {$temp.InventoryDetails[1]/ItemSellingPrice }
Commission Percentage: {$temp.InventoryDetails[1]/SalesCommissionInPercentage }
<br/><br/>
<b>Margins</b>
Overall Profit: {$output.Calculate_Margin_ServiceResponse[1]/MarginBeforeCommission}
Sales Commission: {$output.Calculate_Margin_ServiceResponse[1]/SalesCommission}
Profit after Commission: {$output.Calculate_Margin_ServiceResponse[1]/MarginAfterCommission}
</html>

```

式エディタで次のコンテンツを使用して、HTML コンテンツの変数を定義し、変数をシリアル化して、有効な形式の電子メールを受信します。

```

let $doc :=
<html>
  Order {$output.OrderID} has been submitted for {$input.CustomerEmail}.
  <br/>
  <b>Order details for your records.</b>
  <br/><br/>
  Item Cost: {$temp.InventoryDetails[1]/ItemCostPrice}
  Item Count: {$temp.InventoryDetails[1]/ItemCount }
  Item Sell Price: {$temp.InventoryDetails[1]/ItemSellingPrice }
  Commission Percentage: {$temp.InventoryDetails[1]/SalesCommissionInPercentage }
  <br/><br/>
  <b>Margins</b>
  Overall Profit: {$output.Calculate_Margin_ServiceResponse[1]/MarginBeforeCommission}
  Sales Commission: {$output.Calculate_Margin_ServiceResponse[1]/SalesCommission}
  Profit after Commission: {$output.Calculate_Margin_ServiceResponse[1]/MarginAfterCommission}
</html>
return serialize($doc)

```

- HTML コンテンツに有効な XML が含まれる場合は、式エディタで XQuery 関数 `util:toXML` を使用して、コンテンツを文字列形式にシリアル化します。

例として、HTML コンテンツが次のような場合を考えます。

```

<html>
<head>TaskDetails</head>
<body>
<table>
<tbody>
<tr>
<td>Task Name</td>
<td>Job Id</td>
<td>Status</td>
<td>Start Time</td>
<td>End Time</td>
</tr>
<tr>
<td>Williams</td>
<td>John</td>
<td>{fn:current-date()}</td>
</tr>
</tbody>
</table>
</body>
</html>

```

式エディタで次のコンテンツを使用して、このコンテンツを文字列形式にシリアル化します。

```

util:toXML(<html>
<head>TaskDetails</head>
<body>
<table>
<tbody>
<tr>
<td>Task Name</td>
<td>Job Id</td>
<td>Status</td>
<td>Start Time</td>
<td>End Time</td>
</tr>
<tr>

```

```

        <td>Williams</td>
        <td>John</td>
        <td>{fn:current-date()}</td>
    </tr>
</tbody>
</table>
</body>
</html>

```

- HTML コンテンツに有効な XML が含まれていない場合は、HTML コンテンツを有効な XML に変換する必要があります。HTML コンテンツに有効な XML が含まれていても、有効な形式の電子メールを受信しない場合があります。この場合、String Concat 関数を使用する必要があります。

例として、HTML コンテンツが次のような場合を考えます。

```

<html>
<table>
  <tr>
    <th>Task Property</th>
    <th>Property Value</th>
  </tr>
  <tr>
    <td>Task Name</td>
    <td>{Temp.DataTask1[1]/output[1]/Object_Name}</td>
  </tr>
  <tr>
    <td>Task Status</td>
    <td>{Temp.DataTask1[1]/output[1]/Task_Status}</td>
  </tr>
  <tr>
    <td>Error Message</td>
    <td>{Temp.DataTask1[1]/output[1]/Error_Message}</td>
  </tr>
</table>
</html>

```

式エディタで String Concat 関数を使用して次のコンテンツを使用し、有効な形式の電子メールを送信します。

```

fn:concat(fn:concat(fn:concat(fn:concat(fn:concat("Task Details: <br/><br/>", $temp.DataTask1[1]/
output[1]/Object_Name), "<br/><br/>"), $temp.DataTask1[1]/output[1]/Task_Status), "<br/><br/>"),
$temp.DataTask1[1]/output[1]/Error_Message)

```


- HTML コンテンツに有効な XML が含まれない場合は、HTML コンテンツを有効な XML に変換するか、エスケープ文字を使用して文字列連結関数を使用する必要があります。

次の例は、エスケープ文字と一緒に文字列連結関数を使用する方法を示します。

```
fn:concat("&lt;html&gt;
&lt;head&gt;TaskDetails&lt;/head&gt;
&lt;body&gt;
&lt;table&gt;
&lt;tbody&gt;
&lt;tr&gt;
&lt;td&gt;Task Name&lt;/td&gt;
&lt;td&gt;Job Id&lt;/td&gt;
&lt;td&gt;Status&lt;/td&gt;
&lt;td&gt;Start Time&lt;/td&gt;
&lt;td&gt;End Time&lt;/td&gt;
&lt;/tr&gt;
&lt;tr&gt;
&lt;td&gt;Williams&lt;/td&gt;
&lt;td&gt;John&lt;/td&gt;
&lt;/tr&gt;
&lt;/tbody&gt;
&lt;/table&gt;
&lt;/body&gt;
&lt;/html&gt;", "")
```

コマンドタスクステップ

コマンドタスクステップを使用して、Secure Agent マシン上の複数のスクリプトファイルからシェルスクリプトまたはバッチコマンドを実行します。例えば、コマンドタスクを使用して、ファイルの移動、ファイルのコピー、ファイルの圧縮や解凍、またはクリーンスクリプトや SQL スクリプトの実行をタスクフローの一部として行えます。コマンドタスクの出力を使用して、タスクフローの後続のタスクを調整できます。

コマンドタスクステップを使用するには、適切なライセンスが必要です。

タスクフローにコマンドタスクステップを追加するときに、次のプロパティを設定します。

全般プロパティ

全般プロパティで、コマンドタスクステップにわかりやすい名前を指定できます。

入力フィールド

入力フィールドで次のようなシステム変数を使用して、スクリプトファイル名、入力引数、および作業ディレクトリを定義できます。

- \$PMRootDir
- \$PMLookupFileDir
- \$PMSessionLogDir

- \$PMBadFileDir
- \$PMCacheDir
- \$PMStorageDir
- \$PMTargetFileDir
- \$PMSourceFileDir
- \$PMExtProcDir
- \$PMTempDir
- \$PMWorkflowLogDir

これらの変数は、データ統合サーバーサービス用として Administrator に事前に定義されています。コマンドタスクステップでシステム変数を使用するには、共通の統合コンポーネントおよびデータ統合サーバーサービスが有効になっており、Secure Agent で実行されている必要があります。

入力フィールドで環境変数を使用して、スクリプトファイル名、入力引数、およびワークディレクトリを定義できます。環境変数を使用するには、Secure Agent に共通統合コンポーネントサービスバージョン 14 以降、およびコマンドエグゼキュータパッケージバージョン 140 以降が必要です。

コマンドタスクステップに次の入力フィールドを設定します。

いずれかのスクリプトが失敗すると、タスクも失敗します

複数のスクリプトファイルを使用する場合、いずれかのスクリプトが失敗した場合にコマンドタスクステップが失敗するように設定できます。

このオプションを有効にすると、いずれかのスクリプトが失敗した場合、コマンドタスクステップのステータスが [失敗] に設定され、タスクフローは、それ以上のスクリプトや後続のステップを実行しません。

コマンドタスクステップに単一のスクリプトを設定した場合は、このオプションを選択するかどうかに関係なく、スクリプトが失敗すると、コマンドタスクステップのステータスは [失敗] に設定され、タスクフローは後続のステップを実行しません。

ランタイム環境

必須。コマンドを実行するランタイム環境。この選択は、ランタイム環境またはサーバーレスランタイム環境に適用されます。

次のいずれかのオプションを使用して、このフィールドの値を指定します。

- **コンテンツ。** Secure Agent グループを選択します。
注: サブ組織のユーザーが、使用可能な Secure Agent グループを表示するには、デザイナロールが必要です。
- **フィールド。** このステップを実行するときに、タスクフローデザイナーがこのフィールドにランタイム環境を書き込むために使用するフィールドを選択します。追加されたランタイム環境は、タスクフローの他のステップの入力フィールドまたは一時フィールドとして選択できます。値は、有効な Secure Agent グループ名である必要があります。
- **式。** 有効な Secure Agent グループ名を使用して、ランタイム環境の式を作成します。

スクリプト

コマンドタスクステップでは、複数のスクリプトを追加できます。スクリプトは、スクリプトファイル名、入力引数、および作業ディレクトリを指定することで設定できます。

スクリプトをさらに追加するには、[スクリプト] パネルの **[追加]** をクリックします。コマンドタスクに複数のスクリプトがある場合は、ドラッグアンドドロップでスクリプトを並べ替えることができます。

別の Secure Agent グループからのスクリプトを実行するには、別のランタイム環境に個別のコマンドタスクステップを追加する必要があります。

[すべてのジョブ]、**[実行中のジョブ]**、または **[マイジョブ]** ページから、タスクフローインスタンス内のすべてのスクリプトをコマンドタスクのサブタスクとして表示できます。スクリプトが失敗した場合は、ログファイルをダウンロードして、スクリプト失敗の理由を探ることもできます。

スクリプトの次の入力フィールドを設定します。

スクリプトファイル名

必須。実行するスクリプトファイルのパスおよび名前。

サーバーレスランタイム環境では、ファイルを `command_scripts` という名前のフォルダに配置する必要があります。このフォルダにはサブフォルダを含めることができます。Informatica Intelligent Cloud Services は、`command_scripts` ディレクトリ内のファイルを Secure Agent のエージェントインストールディレクトリ `apps/Common_Integration_Components/data/command/serverless/command_scripts` に定期的に同期します。リモートストレージの場所（Amazon S3 など）でファイルを更新した場合、Informatica Intelligent Cloud Services はそれらのファイルを Secure Agent に自動的に同期します。

次のいずれかのオプションを使用して、このフィールドの値を指定します。

- **コンテンツ**。実行するスクリプトへのパスを入力します。
- **フィールド**。このステップを実行するときに、タスクフローデザイナーがこのフィールドにスクリプトファイル名を書き込むために使用するフィールドを選択します。追加されたスクリプトファイル名は、タスクフローの他のステップの入力フィールド、一時フィールド、または出力フィールドとして選択できます。
- **式**。スクリプトファイルの式を作成します。

システム変数を使用して、実行するスクリプトファイルを定義できます。

例えば、ルートディレクトリにある `script.bat` ファイルを実行するには、`$PMRootDir/script.bat` という値を入力します。

環境変数を使用して、オペレーティングシステムに基づいてスクリプトファイルのパスを定義できます。

例えば、次のディレクトリに `java_script.bat` ファイルがあるとします。

`C:\Scripts`

ディレクトリ用に `ScriptHome` という名前の環境変数を作成しました。`java_script.bat` ファイルを実行するには、Windows の場合は `%ScriptHome%\java_script.bat`、Linux の場合は `$ScriptHome/java_script.bat` のように値を入力します。

また、スクリプトファイル名を含むフルパスの環境変数を作成することもできます。

入力引数

オプション。スクリプトの実行時にスクリプトに渡す引数。

次のいずれかのオプションを使用して、このフィールドの値を指定します。

- **コンテンツ**。スクリプトに渡す 1 つ以上の引数を入力します。各引数は二重引用符 (") で囲み、カンマ (,) を使用して区切ります。
- **フィールド**。このステップを実行するときに、タスクフローデザイナーがこのフィールドに入力引数を書き込むために使用するフィールドを選択します。追加された入力引数は、タスクフローの他のステップの入力フィールド、一時フィールド、または出力フィールドとして選択できます。
- **式**。入力引数の式を作成します。

システム変数を使用して、スクリプトの実行時にスクリプトに渡す入力引数を定義できます。

例えば、ルートディレクトリまたは一時ディレクトリから引数を渡すには、"\$PMRootDir", "\$PMTempDir" という値を入力します。

環境変数を使用して、オペレーティングシステムに基づいて入力引数を定義できます。

例えば、次のディレクトリ用に ScriptHome という名前の環境変数を作成したとします。

C:\Scripts

このディレクトリからの引数を渡すためには、Windows の場合は"%ScriptHome%"、Linux の場合は"\$ScriptHome"のように値を入力します。

作業ディレクトリ

オプション。スクリプトの出力が保存される作業ディレクトリへのパス。デフォルトでは、出力はスクリプトの保存先のパスに保存されます。

サーバーレスランタイム環境では、作業ディレクトリは次のように設定されます: /command_scripts。

次のいずれかのオプションを使用して、このフィールドの値を指定します。

- **コンテンツ**。スクリプトの出力を保存する作業ディレクトリへのパスを入力します。
- **フィールド**。このステップを実行するときに、タスクフローデザイナーがこのフィールドに作業ディレクトリを書き込むために使用するフィールドを選択します。追加された作業ディレクトリは、タスクフローの他のステップの入力フィールド、一時フィールド、または出力フィールドとして選択できます。
- **式**。作業ディレクトリの式を作成します。

システム変数を使用して、スクリプトの出力を保存する作業ディレクトリを定義できます。

例えば、ソースファイルディレクトリを作業ディレクトリとして使用する場合は、\$PMSourceFileDir という値を入力します。

環境変数を使用して、オペレーティングシステムに基づいてワークディレクトリを定義できます。

例えば、次のディレクトリ用に ScriptHome という名前の環境変数を作成したとします。

C:\Scripts

このディレクトリをワークディレクトリとして使用するためには、Windows の場合は%ScriptHome%、Linux の場合は\$ScriptHome のように値を入力します。

注: スクリプトまたは入力フィールドで波括弧 ({}) を使用する場合は、さらに別の波括弧 ({}) を追加する必要があります。これに従っていない場合、エラーが発生します。これは、XQuery では波括弧が特殊文字と見なされるためです。

出力フィールド

タスクフローを実行するときに、次の出力フィールドがコマンドタスクステップ用に生成されます。

出力フィールド	タイプ	説明
実行 ID	整数	コマンドタスクの実行 ID。
開始時刻	日付/時刻	コマンドタスクの開始時刻。
終了時刻	日付/時刻	コマンドタスクの終了時刻。

出力フィールド	タイプ	説明
終了コード	整数	コマンドタスクの実行後に返される終了コード。終了コードは次のいずれかの値を取る場合があります。 - 0: コマンドタスクが正常に実行されたことを示します。 - 1: コマンドタスクが失敗したことを示します。
実行ステータス	テキスト	コマンドタスクのステータスを成功または失敗として表示します。
標準エラー	テキスト	エラーメッセージを表示します。

各出力フィールドの値を表示するには、タスクフローを実行し、Monitor の【**タスクフローインスタンスの詳細**】ページに移動します。【**タスクフローインスタンスの詳細**】ページの詳細については、Monitor のヘルプを参照してください。

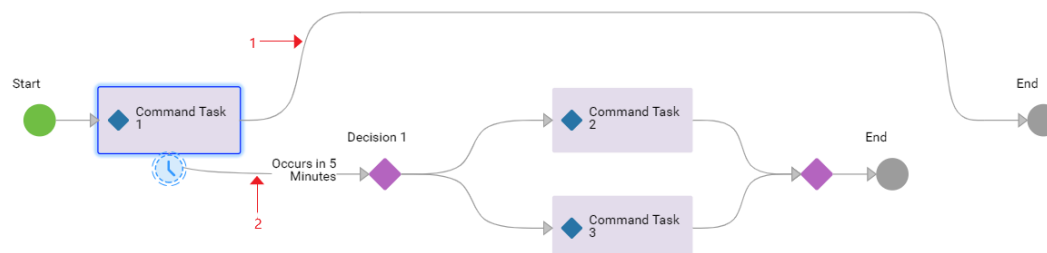
イベント

コマンドタスクにタイマを追加するには、【**イベント**】プロパティを設定します。

タイマイベントを使用して、スケジュールに基づいてアクションを実行します。アクションは、特定の時刻または間隔の後のいずれかになります。

コマンドタスクステップにタイマを追加すると、新しい分岐が表示されます。この分岐にイベントを追加し、特定の【**時間**】または間隔【**以降**】に実行するかどうかを指定します。

次の図では、タイマ分岐のイベントはデータ決定ステップ（決定 1）で、メインコマンドタスク（コマンドタスク 1）開始の 5 分後に発生します。

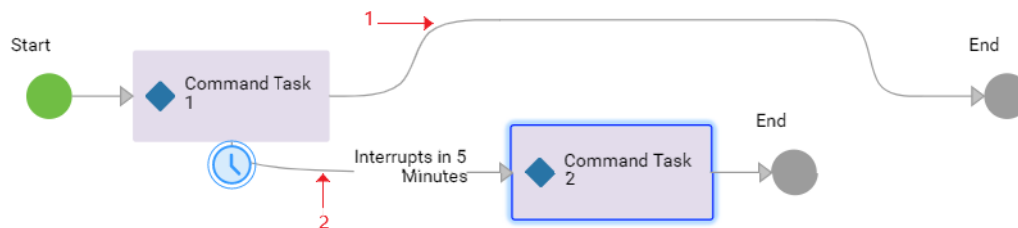


1. メイン分岐
2. タイマ分岐

この例では、タイマ分岐はコマンドタスク 1 が開始されてから 5 分後に実行されます。コマンドタスク 1 が決定 1 の前に終了する場合、タイマ分岐は実行されません。

メインコマンドタスクをタイマで中断する場合は、【**中断**】オプションを選択できます。

次の図は、メインコマンドタスクが開始されてから 5 分後に発生するように設定されている中断タイマを示しています。



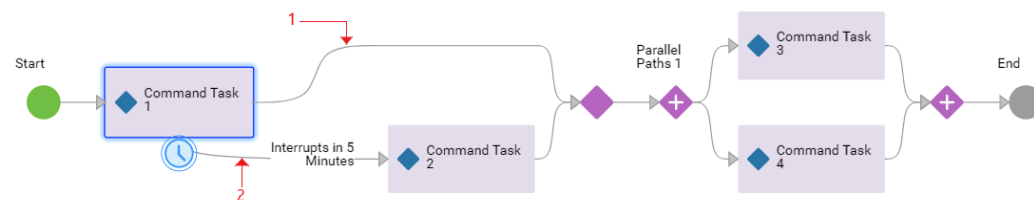
1. メイン分岐
2. タイマ分岐

この例では、コマンドタスク 2 が 5 分後に実行され、コマンドタスク 1 が中断されます。タスクフローでは、コマンドタスク 2 のみが実行されて終了します。コマンドタスク 1 には、このタスクフローインスタンスの出力はありません。タスクフローのジョブの詳細ページには、コマンドタスク 1 の出力フィールドが表示されません。

コマンドタスク 1 がタイマより前に終了した場合、タスクフローはコマンドタスク 1 のみを実行して終了します。

中断タイマのタイマ分岐の終了ステップを削除すると、タイマ分岐はメイン分岐に再び参加します。

次の図は、終了ステップが削除された中断タイマ分岐を示しています。



1. メイン分岐
2. タイマ分岐

この例では、コマンドタスク 2 が 5 分後に実行され、コマンドタスク 1 が中断されます。タイマ分岐は、メイン分岐に再び参加します。タスクフローは、コマンドタスク 2、決定ステップ、並列パスステップを実行し、終了します。

コマンドタスク 1 がタイマより前に終了した場合、タスクフローは、コマンドタスク 1、決定ステップ、並列パスステップを実行し、終了します。

エラー処理

【エラー処理】 タブを使用すると、コマンドタスクステップでエラーが発生したときにタスクフローがどのように動作するかを指定できます。コマンドタスクステップが失敗した場合、または実行されなかった場合のタスクフローの動作を設定することもできます。

タスクを選択した後に、次のエラー処理のプロパティを設定します。

エラー発生時

コマンドタスクステップでエラーが発生したときにタスクフローが取るパス。

コマンドタスクステップが失敗するとエラーが発生します。次のオプションから選択できます。

- **【無視】** を選択してエラーを無視し、次の手順に進みます。
注: 通知タスクステップが後に続くコマンドタスクステップに **【無視】** を選択してコマンドタスクが失敗した場合は、受信する電子メール通知にはフォールトの詳細は含まれません。電子メールでフォールトの詳細を確認するには、**【カスタムエラー処理】** を選択します。
- エラーが発生したときにタスクフローを中断状態に移行させるには、**【タスクフローの中断】** を選択します。**【すべてのジョブ】**、**【実行中のジョブ】**、および**【マイジョブ】** ページからタスクフローインスタンスを再開できます。
タスクフローは、中断されたステップから再開します。エラーの理由がわかっている場合は、問題を修正してからタスクフローを再開してください。
- 選択した方法でエラーを処理するには、**【カスタムエラー処理】** を選択します。**【カスタムエラー処理】** を選択すると、2 つの分岐が表示されます。最初の分岐は、エラーが発生しない場合にタスクフローが続くパスです。2 番目の分岐は、エラーが発生した場合にタスクフローが続くカスタムパスです。

デフォルト値は **【タスクフローの一時停止】** です。

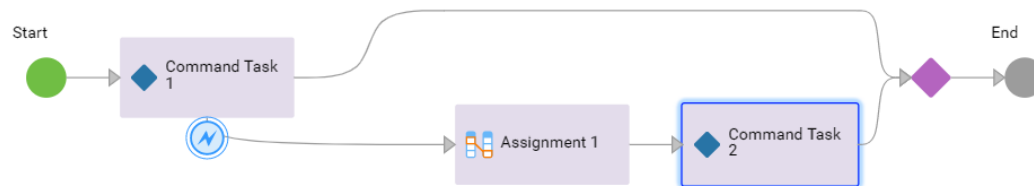
タスクフローを完了時に終了

コマンドタスクステップが失敗した場合、または実行されなかった場合のタスクフローの動作。

コマンドタスクステップが失敗した場合、または実行されなかった場合、完了時に失敗するようにタスクフローを設定できます。ステップが失敗するか実行されなかった場合、タスクフローは後続のステップの実行を続行します。ただし、タスクフローが完了すると、タスクフローのステータスは失敗に設定されます。

【フォールト時に一時停止】 タスクフローの詳細プロパティと **【タスクフローを完了時に終了】** プロパティの両方を設定した場合は、**【フォールト時に一時停止】** プロパティが優先されます。この場合、コマンドタスクステップが失敗するか実行されない場合、タスクフローは一時停止されます。タスクフローは、そのコマンドタスクステップの後の後続のステップを実行しません。

次の図は、割り当てステップと別のコマンドタスクステップがある、**カスタムエラー処理** パスを示しています。



フォールト

タスクフローの実行後、**【ジョブ】** または **【マイジョブ】** ページでタスクフローインスタンスに関する情報の詳細を参照できます。タスクに固有のプロパティを表示するには、コマンドタスクステップをクリックします。

コマンドタスクが失敗した場合は、次のフォールトの詳細を参照できます。

プロパティ	説明
コード	障害がエラーであるか、または警告であるかを示します。
理由	フォールトの理由。
詳細	フォールトの詳細。

File Watch タスクステップ

File Watch タスクステップをタスクフローに追加して、定義された場所にあるファイルをリッスンし、ファイルイベントを監視できます。

File Watch タスクステップでは、コネクタソースタイプを使用して既存のファイルリスナを選択できます。タスクフローの実行を調整するため、ファイルイベントを使用できます。例えば、ファイルが特定の場所に到着するまで待機してから、次のステップでそのファイルを使用するといったことができます。

File Watch タスクステップを含むタスクフローを実行すると、関連するファイルリスナが起動します。File Watch タスクはファイルイベントが発生すると実行され、ファイルの詳細とともに、到着、更新、および削除されたファイルのリストなどのファイルイベントの詳細が送信されます。その後、タスクフローは後続のステップに進みます。

ファイルイベントが発生しない場合、タスクフローは、デフォルトでは 5 分間または **【タイムアウト】** フィールドで定義したオーバーライド値の分だけ待機します。その後、File Watch タスクステップが完了し、タスクフローは後続のステップに進みます。

【タイムアウト】 フィールドで定義できる最大値は 7 日間です。7 日後、タスクフローのステータスは **【一時停止中】** に変わります。

次の節では、File Watch タスクステップのプロパティについて説明します。

全般プロパティ

全般プロパティで、ファイル監視タスクステップにわかりやすい名前を指定できます。

File Watch タスクステッププロパティ

File Watch タスクステッププロパティで、タスクフローに追加するコネクタソースタイプを持った既存のファイルリスナを選択できます。ファイルリスナを選択すると、ファイルリスナの説明と出力フィールドが表示されます。

【出力フィールド】 セクションには **【fileEvents】** フィールドが表示されます。**【fileEvents】** フィールドは、到着、更新、および削除されたファイルのリストやファイルの詳細など、ファイルイベントの詳細を返すオブジェクトのリストです。**【fileEvents】** フィールドに表示されるレコード数は、（処理された合計ファイル数が 5,000 を超えていても）最大 5,000 件です。

入力フィールド

入力フィールドを追加して、次のファイルリスナプロパティをオーバーライドできます。

- 最初の実行時にファイルが存在するかどうかを通知
- ランタイム環境
- ソース接続
- タイムアウト
- ファイルパターン
- フォルダパス

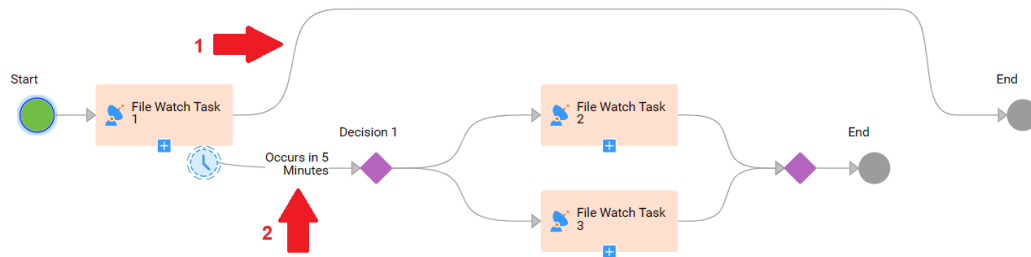
イベント

File Watch タスクにタイマを追加するには、**【イベント】** プロパティを設定します。

タイマイベントを使用して、スケジュールに基づいてアクションを実行します。アクションは、特定の時刻または間隔の後のいずれかになります。

File Watch タスクステップにタイマを追加すると、新しい分岐が表示されます。この分岐にイベントを追加し、特定の **【時間】** または間隔 **【以降】** に実行するかどうかを指定します。

次の図では、タイマ分岐のイベントは決定ステップ（決定 1）で、メインの File Watch タスク（File Watch タスク 1）開始の 5 分後に発生します。

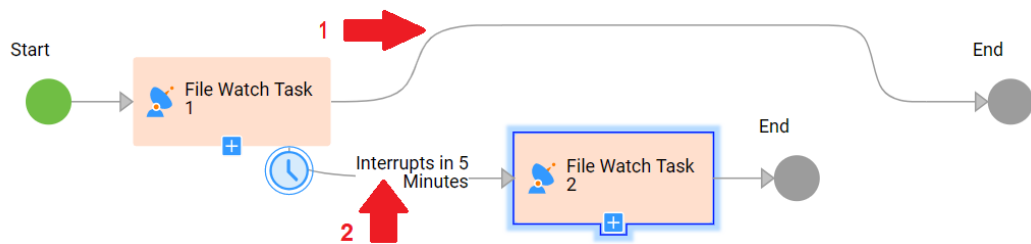


1. メイン分岐
2. タイマ分岐

この例では、タイマ分岐は File Watch タスク 1 が開始されてから 5 分後に実行されます。File Watch タスク 1 が決定 1 の前に終了する場合、タイマ分岐は実行されません。

メインの File Watch タスクをタイマで中断する場合は、**【中断】** オプションを選択できます。

次の図は、メインの File Watch タスクが開始されてから 5 分後に発生する中断タイマ設定を示しています。



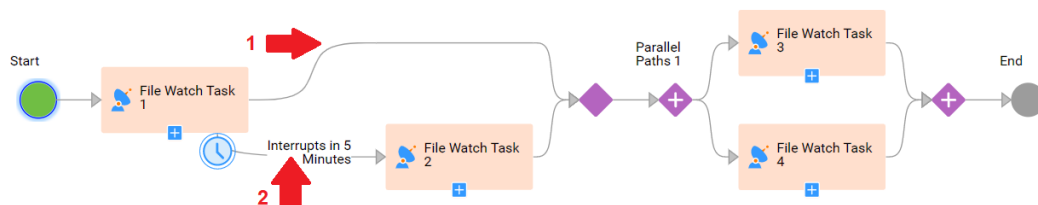
1. メイン分岐
2. タイマ分岐

この例では、File Watch タスク 2 が 5 分後に実行され、File Watch タスク 1 が中断されます。タスクフローでは、File Watch タスク 2 のみが実行されて終了します。File Watch タスク 1 には、このタスクフローインスタンスの出力はありません。タスクフローのジョブの詳細ページには、File Watch タスク 1 の出力フィールドが表示されません。

File Watch タスク 1 がタイマより前に終了した場合、タスクフローは File Watch タスク 1 のみを実行して終了します。

中断タイマのタイマ分岐の終了ステップを削除すると、タイマ分岐はメイン分岐に再び参加します。

次の図は、終了ステップが削除された中断タイマ分岐を示しています。



1. メイン分岐
2. タイマ分岐

この例では、File Watch タスク 2 が 5 分後に実行され、File Watch タスク 1 が中断されます。タイマ分岐は、メイン分岐に再び参加します。タスクフローでは、File Watch タスク 2、決定ステップ、並列パスステップが実行されて終了します。

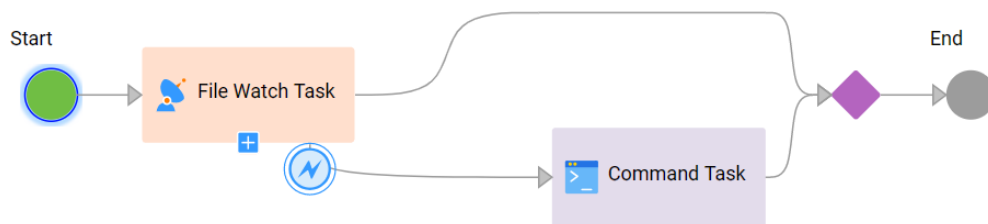
File Watch タスク 1 がタイマより前に終了した場合、タスクフローでは、File Watch タスク 1、決定ステップ、並列パスステップが実行されて終了します。

エラー処理のプロパティ

【エラー処理】 タブを使用すると、File Watch タスクステップがエラーを検出したときにタスクフローがどのように動作するかを指定できます。ファイルリスナを選択した後で、**【エラー発生時】** プロパティを設定できます。

【エラー発生時】 プロパティにより、File Watch タスクステップでエラーが発生した場合にタスクフローがたどるパスを定義します。File Watch タスクステップが失敗すると、エラーが発生します。次のオプションから選択できます。

- **【無視】** を選択してエラーを無視し、次の手順に進みます。
- エラーが発生したときにタスクフローを中断状態に移行させるには、**【タスクフローの中断】** を選択します。**【すべてのジョブ】**、**【実行中のジョブ】**、または **【マイジョブ】** ページからタスクフローインスタンスを再開できます。タスクフローは、中断されたステップから再開します。エラーの理由が分かる場合は、問題を修正してからタスクフローを再開してください。
- 任意の方法でエラーを処理するには、**【カスタムエラー処理】** を選択します。**【カスタムエラー処理】** を選択すると、2 つの分岐が表示されます。最初の分岐は、エラーが発生しない場合にタスクフローがたどるパスです。2 番目の分岐は、エラーが発生した場合にタスクフローがたどるカスタムパスです。次の図は、File Watch タスクステップによるカスタムエラー処理パスを示しています。



デフォルト値は **【タスクフローの一時停止】** です。

取り込みタスクステップ

取り込みタスクステップを使用して、ファイル取り込みタスクをタスクフローの調整に活用します。取り込みタスクステップでは、既存のファイル取り込みタスクを選択できます。

注: ファイルリスナコンポーネントをソースとして使用するファイル取り込みタスクは選択できません。

ファイルを中間位置に移動した後で、ファイルをターゲットに転送する前にデータ統合操作を実行することをお勧めします。この場合、取り込みタスクステップをデータタスクステップと組み合わせて使用できます。

例えば、取り込みタスクステップを使用してソースの場所から多数のファイルを読み取り、それらを中間位置に書き込むことができます。次に、データタスクステップを使用してファイルに対してデータ統合操作を実行し、別の取り込みタスクステップを使用して更新済みファイルを最終的なターゲットの場所へ書き込むことができます。

次の節では、取り込みタスクステップのプロパティについて説明します。

全般プロパティ

全般プロパティで、取り込みタスクステップにわかりやすい名前を指定できます。

取り込みタスクステップのプロパティ

取り込みタスクステップのプロパティで、タスクフローに追加するファイル取り込みタスクを選択できます。ファイル取り込みタスクを選択すると、ファイル取り込みタスクの説明と出力フィールドが表示されます。

【出力フィールド】セクションには、次のフィールドが表示されます。

プロパティ	説明
成功ファイル	ターゲットに正常に転送されたファイルの数。
エラーファイル	ターゲットに転送されなかったファイルの数。
ファイルの詳細	作成、更新、および削除されたファイルのリストやファイル数など、ファイル転送の詳細を返すオブジェクトのリスト。【fileDetails】フィールドに表示されるレコード数は、（処理された合計ファイル数が5,000を超えていても）最大 5,000 件です。

入力フィールド

入力フィールドを追加して、次のファイル取り込みタスクのプロパティをオーバーライドできます。

- 全般プロパティ。
 - ランタイム環境
 - ソース接続
 - ターゲット接続
- ソースのプロパティ。
 - バッチサイズ
 - ファイルパターン
 - フォルダパス
 - ソースディレクトリ
- ターゲットプロパティ。
 - フォルダパス
 - ターゲットディレクトリ

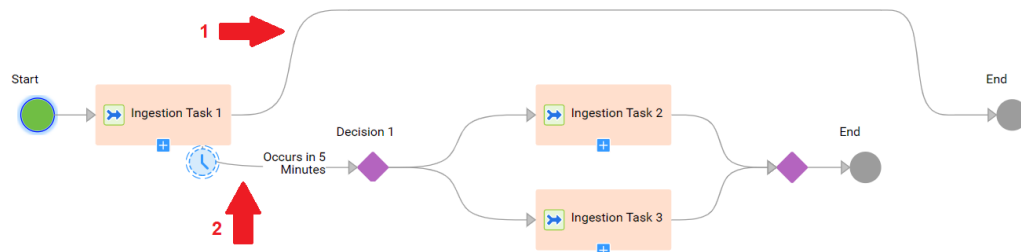
イベント

取り込みタスクにタイマを追加するには、【イベント】プロパティを設定します。

タイマイベントを使用して、スケジュールに基づいてアクションを実行します。アクションは、特定の時刻または間隔の後のいずれかになります。

取り込みタスクステップにタイマを追加すると、新しい分岐が表示されます。この分岐にイベントを追加し、特定の【時間】または間隔【以降】に実行するかどうかを指定します。

次の図では、タイマ分岐のイベントは決定ステップ（決定 1）で、メインの取り込みタスク（取り込みタスク 1）開始の 5 分後に発生します。

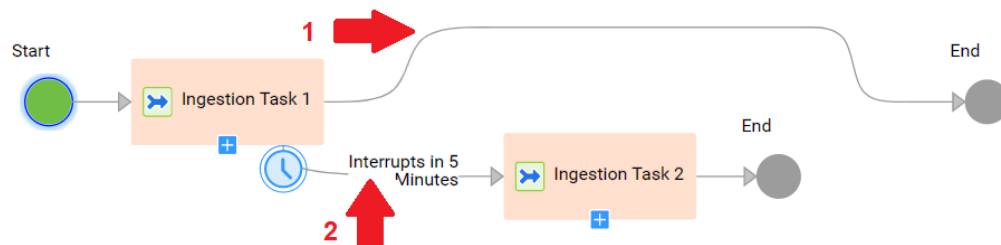


1. メイン分岐
2. タイマ分岐

この例では、タイマ分岐は取り込みタスク 1 が開始されてから 5 分後に実行されます。取り込みタスク 1 が決定 1 の前に終了する場合、タイマ分岐は実行されません。

メインの取り込みタスクをタイマで中断する場合は、**[中断]** オプションを選択できます。

次の図は、メインの取り込みタスクが開始されてから 5 分後に発生する中断タイマ設定を示しています。



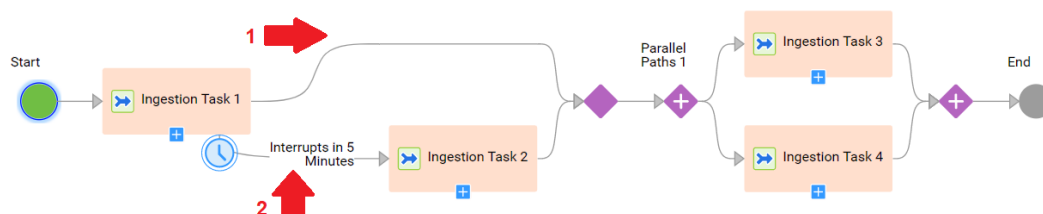
1. メイン分岐
2. タイマ分岐

この例では、取り込みタスク 2 が 5 分後に実行され、取り込みタスク 1 が中断されます。タスクフローでは、取り込みタスク 2 のみが実行されて終了します。このタスクフローインスタンスでは、取り込みタスク 1 の出力はありません。タスクフローのジョブの詳細ページには、取り込みタスク 1 の出力フィールドが表示されません。

取り込みタスク 1 がタイマより前に終了した場合、タスクフローは取り込みタスク 1 のみを実行して終了します。

中断タイマのタイマ分岐の終了ステップを削除すると、タイマ分岐はメイン分岐に再び参加します。

次の図は、終了ステップが削除された中断タイマ分岐を示しています。



1. メイン分岐
2. タイマ分岐

この例では、取り込みタスク 2 が 5 分後に実行され、取り込みタスク 1 が中断されます。タイマ分岐は、メイン分岐に再び参加します。タスクフローでは、取り込みタスク 2、決定ステップ、並列パスステップが実行されて終了します。

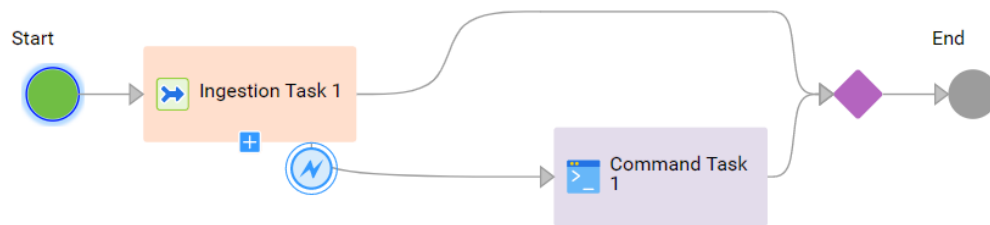
取り込みタスク 1 がタイマより前に終了した場合、タスクフローでは、取り込みタスク 1、決定ステップ、並列パスステップが実行されて終了します。

エラー処理のプロパティ

【エラー処理】タブを使用すると、取り込みタスクステップがエラーを検出したときにタスクフローがどのように動作するかを指定できます。ファイル取り込みタスクを選択した後で、【エラー発生時】プロパティを設定できます。

【エラー発生時】プロパティにより、取り込みタスクステップでエラーが発生した場合にタスクフローがたどるパスを定義します。取り込みタスクステップが失敗するとエラーが発生します。次のオプションから選択できます。

- 【無視】を選択してエラーを無視し、次の手順に進みます。
- エラーが発生したときにタスクフローを中断状態に移行させるには、【タスクフローの中断】を選択します。【すべてのジョブ】、【実行中のジョブ】、または【マイジョブ】ページからタスクフローインスタンスを再開できます。タスクフローは、中断されたステップから再開します。エラーの理由が分かる場合は、問題を修正してからタスクフローを再開してください。
- 任意の方法でエラーを処理するには、【カスタムエラー処理】を選択します。【カスタムエラー処理】を選択すると、2 つの分岐が表示されます。最初の分岐は、エラーが発生しない場合にタスクフローがたどるパスです。2 番目の分岐は、エラーが発生した場合にタスクフローがたどるカスタムパスです。次の図は、取り込みタスクステップによるカスタムエラー処理パスを示しています。



デフォルト値は【タスクフローの一時停止】です。

サブタスクフローステップ

サブタスクフローステップを追加する際は、既存のタスクフローを埋め込んで再利用できます。

サブタスクフローステップを使用すると、タスクフローの複数のブランチ間または異なるタスクフロー間で同じオーケストレーションフローを再利用できます。その後、異なるパラメータセットでタスクフローを呼び出すことができます。サブタスクフローは、親タスクフローをパブリッシュするときにパブリッシュされます。

多数のステップを含むタスクフローがある場合は、オーケストレーションロジックを複数の小さなタスクフローに分割することを検討してください。次に、サブタスクフローステップを使用して、より小さなタスクフローを親タスクフローに組み込むことにより、設計を簡略化できます。これは、モジュラー型設計につながるだけでなく、編集用にタスクフローを開いたときの読み込みを高速化するのにも役立ちます。

サブタスクフローのプロパティは定義できます。次の節では、サブタスクフローステップのプロパティについて説明します。

全般プロパティ

全般プロパティで、サブタスクフローにわかりやすい名前を指定できます。

サブタスクフローのプロパティ

【サブタスクフロー】タブで、埋め込むタスクフローを選択します。データ統合は、埋め込まれたタスクフローの場所、説明、入力フィールド、および出力フィールドを表示します。

入力フィールド

サブタスクフローをタスクフローに追加すると、**【入力フィールド】** セクションが表示されます。

上書きできるパラメータがサブタスクフローに含まれている場合は、入力フィールドを追加できます。入力フィールドのプロパティを設定して、Data Integration のランタイムパラメータを上書きできます。

フォールト処理のプロパティ

次のフォールト処理のプロパティを設定できます。

フォルトの取得

サブタスクフローステップのフォールト処理を有効にするには、このオプションを選択します。

デフォルトでは、このオプションは選択されていません。

フォルトフィールド名

【フォルトの取得】 オプションを選択した場合は必須です。

フォールト情報を取得するフィールドの名前です。

デフォルトは **faultInfo** です。

タスクフローを完了時に終了

サブタスクフローステップに関連付けられたサブタスクフローが失敗または実行されない場合の動作を定義します。

デフォルトでは、サブタスクフローステップに関連付けられたサブタスクフローが失敗すると、親タスクフローも失敗します。

サブタスクフローが失敗した場合に、タスクフローも完了時に失敗するように設定するには、**【フォルトの取得】** オプションと **【このサブタスクフローが失敗した場合】** オプションを選択します。サブタスクフローが実行されない場合に、タスクフローも完了時に失敗するように設定するには、**【このサブタスクフローが実行されない場合】** オプションを選択します。これらの場合、サブタスクフローが失敗するか実行されなくても、タスクフローは後続のステップの実行を続行します。ただし、タスクフローが完了すると、タスクフローのステータスは失敗に設定されます。

注: **【フォールト時に一時停止】** タスクフローの詳細プロパティと **【タスクフローを完了時に終了】** プロパティの両方を設定した場合は、**【フォールト時に一時停止】** プロパティが優先されます。この場合、サブタスクフローステップに関連付けられたサブタスクフローが失敗するか実行されない場合、タスクフローは一時停止されます。タスクフローは、そのサブタスクフローステップの後の後続のステップを実行しません。

決定ステップ

決定ステップを追加するときに、いくつかのプロパティを設定します。

次の決定ステップのプロパティを設定できます。

タイトル

決定ステップの名前。

決定

タスクフローは、ここで定義したフィールドとパスに基づいて決定を行います。

【スタート】 > **【フィールド】** で定義したフィールドの一覧からフィールド名を選択します。

決定ステップで決定を下すために必要な条件と値を入力します。

使用できる条件は、選択したフィールドによって異なります。

例えば、**【簡易】** > **【テキスト】** を選択した場合、次の条件が使用できます。

- 次の値に等しい
- 次で開始する
- 次で終わる
- 次のいずれかで開始する
- 次を含む

選択した条件に対してテキスト値を入力できます。

決定ステップには複数の条件を追加できます。各条件は、潜在的なデータパスです。

追加したパスごとに、対応する分岐が UI に表示されます。分岐をドラッグして、分岐が UI に表示される順序を並べ替えます。

ほとんどの決定ステップには、別のパスがあります。テストの条件を満たしているデータがない場合、このパスが実行を処理します。

パスの評価

タスクフローは、指定した条件に基づいて条件を評価します。パスは必ず、交差しない条件を指定して作成します。

例えば、次のパスでデータ決定ステップを作成します。

- パス 1: フィールドは 100 以下とする。
- パス 2: フィールドは 75 以下とする。
- パス 3: フィールドは 25 以下とする。
- パス 4: それ以外。

データ決定ステップが作成された整数フィールドの値が 25 の場合、データ決定ステップはパス 1 を受け取ります。これは 25 が 100 未満で、パス 1 が最初のオプションであるためです。

データ決定ステップが「25 未満のフィールド以下」のパスに従っていることを確認するには、次の条件でパスを再作成します。

- パス 1: 0 から 25 の間の整数。
- パス 2: 26 から 75 の間の整数。
- パス 3: 76 から 100 の間の整数。
- パス 4: それ以外。

重要: タスクフローは、トップダウン方式で条件を評価します。それ以外の分岐が最後のパスであることを確認します。

決定ステップは、別の決定ステップに連結できます。例えば、年間所得が 10 万ドルを超えると、分岐を実行できるなどと指定できます。同じパスの次の決定テストで、都市がボストンであるか、それ以外であるかをテストできます。この方法を使用するときには、最初の条件での真の分岐に基づいて 2 番目の条件をテストするため、ブール値 AND ロジックを使用します。この例では、決定ステップを使用して、「年間収入が 10 万ドルを超え、都市はボストンである」という AND 条件を設定します。

同様に、ブール型 OR ロジックをサポートするために、任意の分岐に 2 番目の条件のテストを追加できます。

タスクフローのデータタスクステップが失敗した場合は、データタスクの出力フィールドに基づいて決定を下すことができます。

次のいずれかの条件が満たされた場合に、出力フィールドを選択できます。

- **【エラー時】** フィールドが **【無視】** または **【カスタムエラー処理】** に設定されている。
- **【タスクフローを完了時に終了】** オプションが **【このタスクが失敗した場合】** に設定されている。

フィールドをデータタスク全体として選択した場合、決定ステップはデフォルトで、**【設定されている】** パスを取ります。

並列パスステップ

並列パスステップを追加するときに、いくつかのプロパティを設定します。

次の並列パスステップのプロパティを設定できます。

名前

並列パスステップの名前。

並列パス

タスクフローを並列で実行するパスを選択します。

【追加】 をクリックして新しい分岐を追加します。

各分岐に複数のステップを追加できます。分岐にステップを追加するには、左側のパレットからステップをドラッグアンドドロップします。

マッピングタスクが同時に実行するように設定されている場合、同じマッピングタスクを並列パスステップの複数の分岐で実行できます。

ジャンプステップを並列パスステップと連動させて使用した場合は、同じ並列パスブランチ内の別のステップにのみジャンプできます。

ジャンプステップと並列パスステップを同時に使用する場合は次の制限を念頭に置いてください：

- 並列パスステップ上にいる場合は、同じ並列パスステップの別のブランチのステップにジャンプすることはできません。
- 並列パスステップ上にいる場合は、並列パスステップ外のいかなるステップへもジャンプできません。
- 並列パスステップの外にいる場合は、並列パスステップ内のいかなるステップへもジャンプすることはできません。

ジャンプステップ

ジャンプステップを追加するときは、ジャンプのターゲットを定義する **【宛先】** フィールドを設定します。使用可能なステップのリストから選択できます。

複数のステップが同じターゲットステップにジャンプできます。ターゲットとして特定のステップがあるジャンプステップの数を確認するには、ターゲットステップの横にある矢印の上にカーソルを置きます。

ジャンプステップを並列パスステップと連動させて使用した場合は、同じ並列パスブランチ内の別のステップにのみジャンプできます。

ジャンプステップと並列パスステップを同時に使用する場合は次の制限を念頭に置いてください：

- 並列パスステップ上にいる場合は、同じ並列パスステップの別のブランチのステップにジャンプすることはできません。
- 並列パスステップ上にいる場合は、並列パスステップ外のいかなるステップへもジャンプできません。
- 並列パスステップの外にいる場合は、並列パスステップ内のいかなるステップへもジャンプすることはできません。

終了ステップ

終了ステップは、タスクフローの終了を示します。実行がこのステップに達すると、タスクフローは完了します。

次の終了ステップのプロパティを設定できます。

タイトル

ステップの名前。この値は編集できます。

終了タイプ

デフォルト値は End of Process です。この値は編集できません。

HTTP ステータス

HTTP 応答のステータスコード。デフォルト値は 200 OK です。この値は編集できます。

待機ステップ

待機ステップを追加するときに、いくつかのプロパティを設定します。

次の待機ステップのプロパティを設定できます。

名前

待機ステップの名前。

待機

タスクフローが一時停止する時刻と長さを決定するプロパティ。

次の条件を使用して、タスクフローを【特定の時間】または【待機期間後】に一時停止するかどうかを決定します。

- 【特定の時間】を選択して、特定の時刻にタスクフローを一時停止します。タスクフローを一時停止し、必要に応じて【遅延】を指定する【時間】を入力します。【遅延】の値には、定義した整数またはフィールドを指定できます。
例えば、3 日後の午前 2:00 時に一時停止するようにタスクフローを設定するとします。2:00am は【時間】であり、3 日は【遅延】です。
- 期間後にタスクフローを一時停止するには、【待機期間後】を選択します。この期間は、タスクフローが待機ステップに到達した時点で開始されます。タスクフローを一時停止する【待機期間】を入力します。【待機期間】の値には、定義した整数またはフィールドを指定できます。
例えば、タスクフローが待機ステップに到達した時点から 1 時間、タスクフローを一時停止するように設定したりできます。

スローステップ

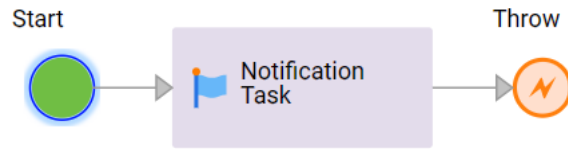
スローステップを使用して、フォールトをキャッチし、フォールトの詳細を返し、タスクフローの実行を停止し、タスクフローのステータスを【失敗】に設定します。

スローステップは次のケースに使用できます。

タスクフロー内でフォールトをキャッチする

スローステップをタスクフローのメインパスに追加することで、タスクフロー内でフォールトをキャッチし、フォールトの詳細を返すことができます。スローステップは中断ステップであるため、スローステップの後にステップを追加することはできません。フォールトが発生すると、スローステップはタスクフローの実行を停止し、タスクフローのステータスを【失敗】に設定します。

例えば、次のサンプルタスクフローについて考えてみます。



通知タスクステップ時にフォールトが発生すると、スローステップが実行されます。スローステップはタスクフローの実行を停止し、タスクフローのステータスを「失敗」に設定します。

タスクフローの特定のステップに対し境界イベントとして機能させる

タスクフローステップに対してカスタムエラー処理を有効にすると、エラー処理パス内でスローステップを使用して、ステップへの境界イベントとして機能させることができます。境界イベントは、境界イベントが定義されたステップの範囲内で生じたエラーをキャッチするイベントです。

スローステップはカスタムエラー処理をサポートしているため、次のステップのエラー処理パスに追加できます。

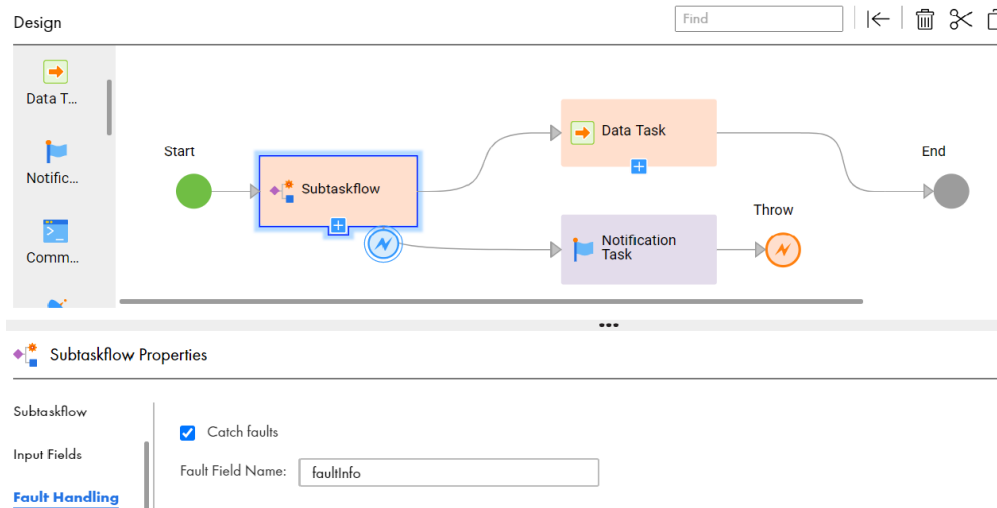
- データタスク
- コマンドタスク
- File Watch タスク
- 取り込みタスク

サブタスクフローステップでフォールトをキャッチするように設定してある場合は、スローステップをサブタスクフローステップのエラー処理パスに追加することもできます。サブタスクフローステップ内にあるタスクフローが失敗すると、親タスクフローも失敗します。Monitor で親タスクフローの実行の詳細を表示する場合は、サブタスクフローステップに関連付けられたスローステップをクリックして、サブタスクフローが失敗した理由を確認できます。

スローステップをエラー処理パスに追加すると、エラー処理パスがタスクフローのメインパスから分岐します。フォールトが発生すると、タスクフローは、エラー処理用に定義したパスに進みます。例えば、エラーパス内に通知タスクステップを追加して電子メール通知を送信し、続いてスローステップでフォールトをキャッチし、タスクフローの実行を停止することができます。

スローステップは中断ステップであるため、エラー処理パス内でスローステップの後にステップを追加することはできません。フォールトが発生すると、スローステップはタスクフローの実行を停止し、タスクフローのステータスを「失敗」に設定します。スローステップに関連付けられたステップ後に存在する、タスクフローのメインパス内の後続ステップは実行されません。

例えば、次のサンプルタスクフローについて考えてみます。



サブタスクフローステップは、フォールトをキャッチするように設定されています。フォールトが発生すると、通知タスクステップで設定されているとおりに電子メール通知が送信されます。その後、スローステップはタスクフローの実行を停止し、サブタスクフローが失敗した理由を示すフォールトの詳細を返し、タスクフローのステータスを「失敗」に設定します。データタスクステップは実行されません。

スローステップのプロパティ

次のセクションでは、スローステップのプロパティについて説明します。

全般プロパティ

全般プロパティで、スローステップにわかりやすい名前を指定できます。

フォールトのフィールド

次のフォールトフィールドを設定できます。

コード

必須。フォールトのコードを定義します。

次のいずれかのオプションを使用して、このフィールドの値を指定します。

- **コンテンツ**。フォールトのコードを入力します。
- **フィールド**。このステップを実行するときに、タスクフローデザイナーがこのフィールドにコードを書き込むために使用するフィールドを選択します。タスクフローの他のステップで追加された入力フィールド、一時フィールド、出力フィールド、またはフォールトフィールドを選択できます。
- **式**。式エディタを開いて、このフィールドの値を計算する式を指定します。

デフォルトは【コンテンツ】です。

詳細

フォールトの詳細を定義します。

次のいずれかのオプションを使用して、このフィールドの値を指定します。

- **コンテンツ**。フォールトの詳細を入力します。

- **フィールド**。このステップを実行するときに、タスクフローデザイナーがこのフィールドにフォールトの詳細を書き込むために使用するフィールドを選択します。タスクフローの他のステップで追加された入力フィールド、一時フィールド、出力フィールド、またはフォールトフィールドを選択できます。
- **式**。式エディタを開いて、このフィールドの値を計算する式を指定します。

デフォルトは **【コンテンツ】** です。

理由

フォールトが発生した理由を定義します。

次のいずれかのオプションを使用して、このフィールドの値を指定します。

- **コンテンツ**。フォールトが発生した理由を入力します。
- **フィールド**。このステップを実行するときに、タスクフローデザイナーがこのフィールドにフォールトの理由を書き込むために使用するフィールドを選択します。タスクフローの他のステップで追加された入力フィールド、一時フィールド、出力フィールド、またはフォールトフィールドを選択できます。
- **式**。式エディタを開いて、このフィールドの値を計算する式を指定します。

デフォルトは **【コンテンツ】** です。

ランタイムパラメータ

次の方法でパラメータを設定できます。

パラメータ

パラメータは、マッピングまたは PowerCenter タスクでの値を表すプレースホルダです。タスクフローを使用して、タスクに入力パラメータと入出力パラメータを渡すことができます。

パラメータファイル

パラメータファイルとは、ユーザー定義パラメータ、およびそれらに関連する値のリストです。

パラメータファイルにタスクフローおよびマッピングタスクのセクションとパラメータが含まれている場合は、データ統合ジョブが実行される場所にパラメータファイルを保存できます。パラメータファイルの詳細については、「[Parameter files](#)」のドキュメントを参照してください。

パラメータセット

パラメータセットとは、ユーザー定義パラメータとそれに関連する値のリストです。

パラメータファイルにタスクフローレベルのセクションとパラメータが含まれている場合は、ParamSetCli ユーティリティを使用して、Informatica が管理するクラウドホステッドリポジトリにパラメータファイルをアップロードし、それらをタスクフローで使用できます。アップロードされたこのパラメータファイルは、パラメータセットと呼ばれます。パラメータセットでは、サブタスクフローとデータタスクステップのパラメータを定義することもできます。

パラメータファイルをパラメータセットに変換するには、ParamSetCli ユーティリティをインストールする必要があります。

例えば、2 つのマッピングタスクを調整するタスクフローがあり、タスクフローの入力と、マッピングタスクのパラメータをパラメータファイルで指定するとします。この場合は、各アセットのセクションを含むパラメータファイルを作成し、データ統合タスクが実行される場所にそのパラメータファイルを保存できます。タスクフローでこのパラメータファイルを使用するには、ParamSetCli を使用して、クラウドホステッドリポジトリにパラメータファイルをアップロードする必要があります。アップロードされたパラメータファイルは、タスクフローのパラメータセットになります。その後、タスクフローでパラメータセット名を指定できます。

さらに、パラメータ化した接続パスワードを使用したマッピングタスクなどのデータ統合タスクがある場合は、機密情報を含むパラメータファイルを別途作成することをお勧めします。タスクフロー関連の情報を含む別のパラメータセットを作成できます。

タスクフローのパラメータ

タスクフローを使用して、タスクに入力パラメータと入出力パラメータを渡すことができます。

入力パラメータまたは入出力パラメータを使用して、マッピングを設計できます。タスクフローにマッピングタスクを追加すると、パラメータ値を上書きできます。マッピングタスクは、これらのパラメータをマッピングに渡します。パラメータ化されたマッピングタスクは、さまざまなシナリオで使用できます。

PowerCenter タスクが入力パラメータまたは入出力パラメータを使用する場合、タスクフローのパラメータを上書きできます。

次のセクションでは、入力および入出力パラメータと、それらをタスクフロー内で使用する方法について説明します。

入力パラメータ

入力パラメータは、マッピングまたは PowerCenter タスクの 1 つまたは複数の値のプレースホルダです。マッピングタスクまたは PowerCenter タスクの設定時に、パラメータの値を定義します。入力パラメータの詳細については、「マッピング」および「タスク」を参照してください。

PowerCenter タスクが入力パラメータを使用する場合、タスクフローの入力パラメータを上書きできます。

タスクフローを使用すると、次のマッピング入力パラメータのサブセットを上書きできます。

- 文字列。マッピングタスクの入力として使用する文字列値を変更します。
- ソースオブジェクト。マッピングタスクが読み取るオブジェクトを変更します。
- ソース接続。マッピングタスクがソースからの読み取りに使用する接続を変更します。
- ターゲット接続。マッピングタスクがターゲットへの書き込みに使用する接続を変更します。
- ターゲットオブジェクト。マッピングタスクが書き込むオブジェクトを変更します。

例えば、完全にパラメータ化されたマッピングタスクについて考えてみます。マッピングタスクでは、SQL 接続を使用して `employeeaddress` テーブルから読み取りを行い、JDBC 接続を使用して `employeedetails` テーブルに書き込みます。

タスクフローを作成し、マッピングタスクでパラメータを上書きすることができます。例えば、Salesforce 接続を使用して、`employeeincome` テーブルとフラットファイル接続から読み取りを行い、ファイル `income.txt` に書き込むことができます。

入出力パラメータ

入出力パラメータは、マッピングまたは PowerCenter タスクの内外に渡すことができる値のプレースホルダです。タスクフローを使用すると、マッピングタスクまたは PowerCenter タスクがサポートするすべてのタイプの入出力パラメータを上書きできます。入出力パラメータの詳細については、「マッピング」および「タスク」を参照してください。

例えば、`lastprocessedindex` という入出力パラメータを使用して処理する行数を追跡するマッピングを考慮します。マッピングは、実行するたびにインデックスからの処理を再開します。さらに、マッピングは実行するたびに 5000 行を処理するとします。

マッピングタスクが例えばレコード数 50000 に達するまで実行するようにタスクフローを設定できます。

タスクフローでのパラメータの上書き

タスクフローでの使用時にマッピングタスクのパラメータを上書きします。パラメータをデータタスクステップまたは割り当てステップでオーバーライドします。

データタスクステップでのパラメータまたはパラメータファイルの上書き

データタスクステップで PowerCenter タスクまたはマッピングタスクを使用する場合は、タスクのパラメータ値を上書きできます。データタスクステップでマッピングタスクを使用する場合は、マッピングタスクのパラメータファイルディレクトリとパラメータファイル名を上書きできます。

データタスクステップを使用してデータ統合のパラメータまたはパラメータファイルを上書きするには、次の手順を実行します。

1. タスクフローを作成し、データタスクステップを追加します。
2. パラメータを含むタスクまたはパラメータファイルを使用するタスクをデータタスクステップに追加します。
3. **【データタスク】** > **【入力フィールド】** に移動します。
4. **【追加】** をクリックします。
5. 次のいずれかの手順を実行します。
 - a. マッピングタスクのパラメータファイルディレクトリまたはパラメータファイル名を上書きするには、**【TaskProperties パラメータ】** リストを展開します。次に、**【パラメータファイルディレクトリ】** または **【パラメータファイル名】** を選択します。
 - b. マッピングタスクまたは PowerCenter タスクの入力パラメータまたは入出力パラメータを上書きするには、**【入力パラメータ】** リストまたは **【入出力パラメータ】** リストを展開します。次に、上書きするパラメータに移動して選択します。
6. **【編集】** をクリックします。
【値の編集】 ダイアログボックスが表示されます。
7. **【ソース】** で、**【コンテンツ】** を選択します。高度な使用例については、**【フィールド】** または **【数式】** を選択します。
8. デフォルト値を上書きする新しいパラメータファイルディレクトリ、パラメータファイル名、パラメータ、ルックアップオブジェクト、またはルックアップ条件の値を **【値】** に入力します。
9. **【OK】** をクリックします。

割り当てステップによるパラメータの上書き

データ統合パラメータを上書きするための割り当てステップを使用するには、次の手順を実行します。

1. タスクフローを作成し、データタスクステップを追加します。
2. データタスクステップへの入力パラメータを含むマッピングタスクを追加します。
3. **割り当て** ステップをキャンバス上にドラッグします。
4. **【割り当てのプロパティ】** > **【割り当て】** に移動します。
5. **【追加】** アイコンをクリックし、上書きするパラメータに移動して選択します。
6. **【値】** で、**【コンテンツ】** を選択します。高度な使用例については、**【フィールド】** または **【数式】** を選択します。
7. デフォルトのオブジェクトまたは接続を上書きするために使用するオブジェクトまたは接続を入力または選択します。

タスクフローでのパラメータ使用のガイドラインおよびベストプラクティス

続くセクションでは、タスクフローでパラメータをオーバーライドするときのガイドラインとベストプラクティスについて説明しています。

タスクフローでの入力パラメータ使用のガイドライン

タスクフローの入力パラメータを使用するには、次のガイドラインを使用します。

- 入力パラメータをオーバーライドするには、データタスクステップを使用します。ただし、高度なケースでは、割り当てステップで入力パラメータを上書きできます。
- 割り当てステップとデータタスクステップの両方で同じパラメータを上書きするようにフィールドを定義すると、タスクフローはデータタスクステップで割り当てられた値を考慮します。
- データタスクステップを使用して値を入力または入出力パラメータに割り当てる場合、割り当ては相互に独立している必要があります。割り当てステップを使用して値を入力または入出力パラメータに割り当てる場合、割り当て操作は同じステップの前の割り当て操作の結果を使用できます。

例えば、2つのソースオブジェクト SO1 と SO2 があります。SO1 は値 Account でオーバーライドし、SO2 は値 SO1 でオーバーライドします。パラメータをオーバーライドするには、割り当てステップを使用します。

以下の図は、ソースオブジェクト **SO2** の値 **SO1** でのオーバーライドを示しています。

Field	Assigned Using	From
Data Task 1 > Input Parameters > Source > Source 1 > SO1	Content	Account
Data Task 1 > Input Parameters > Source > Source 2 > SO2	Content	{ \$input.DataTask1[1] }/imj

- タスクフローを API として実行する場合、接続パラメータを使用して、実行時に接続名または接続 ID を渡すことができます。

例えば、接続名を使用する場合は、次のサンプルに示すように、接続パラメータに接続名を渡す必要があります。

```
"source":{
  "Source":{
    "object": "abc",
    "connection": "FileSourceConnectionName"
  }
}
```

接続 ID を使用する場合は、次のサンプルに示すように、接続パラメータで接続 ID を渡す必要があります。

```
"source":{
  "Source":{
    "object": "abc",
    "connection": "0100000B00000000000004"
  }
}
```

入出力パラメータの使用のガイドライン

タスクフローの入出力パラメータを使用するには、次のガイドラインを使用します。

- 割り当てステップとデータタスクステップの両方で同じパラメータを上書きするようにフィールドを定義すると、タスクフローはデータタスクステップで割り当てられた値を考慮します。

- ユースケースが特に割り当てステップの使用を必要としない限り、データタスクステップを使用して入出力パラメータを上書きします。
- 入出力パラメータの現在の値は、タスクフローがデータタスクステップを実行してからでないと取得できません。タスクフローがデータタスクステップを実行する前にこの値を入手することはできません。

例: データタスクステップを使用したパラメータの上書き

マッピングタスクで次の入力パラメータを上書きするには、**[MyMT]** をクリックします。

- ソースデータオブジェクトパラメータ MySourceObject (デフォルト値 **input.txt**)。
- ソース接続パラメータ MySourceConnection (デフォルト値 **[マイファイル接続]**)。
- ターゲット接続パラメータ MyTargetConnection (デフォルト値 **[マイファイル接続]**)。
- ターゲットデータオブジェクトパラメータ MyTargetObject (デフォルト値 **output5.txt**)。

[MyMT] は、**[マイファイル接続]** を使用して **input.txt** ファイルを読み取り、**[マイファイル接続]** を使用して **output5.txt** ファイルに書き込みます。

[MyMT] で、**[マイファイル接続]** を使用して **input.txt** ファイルから読み取り、テーブルを使用し、それから **[マイファイル接続]** を使用して別の出力ファイルである **output10.txt** ファイルに書き込むことができます。

これを行うには、次の手順を実行して、データタスクステップを使用して MyTargetObject のデフォルト値を上書きします。

1. タスクフローを作成し、データタスクステップ **[データタスク 1]** を追加します。
2. **[データタスク 1]** > **[データタスク]** に移動して、マッピングタスク **[MyMT]** を追加します。
3. **[データタスク 1]** > **[入力フィールド]** に移動します。
4. 次の手順を実行して、MySourceConnection を **[データタスク 1]** の **[入力フィールド]** セクションに追加します。
 - a. **[追加]** アイコンをクリックし、MySourceConnection に移動して選択します。
 - b. **[編集]** をクリックします。**[値の編集]** ダイアログボックスが開きます。
 - c. **[ソース]** の横のフィールドで、**[コンテンツ]** を選択します。
 - d. **[値]** の横のフィールドで、**[マイファイル接続]** を選択します。
5. 次の手順を実行して、MySourceObject を **[データタスク 1]** の **[入力フィールド]** セクションに追加します。
 - a. **[追加]** アイコンをクリックし、MySourceObject に移動して選択します。
 - b. **[編集]** をクリックします。**[値の編集]** ダイアログボックスが開きます。
 - c. **[ソース]** の横のフィールドで、**[コンテンツ]** を選択します。
 - d. **[値]** の横のフィールドで、**input.txt** と入力します。
6. 次の手順を実行して、MyTargetConnection を **[データタスク 1]** の **[入力フィールド]** セクションに追加します。
 - a. **[追加]** アイコンをクリックし、MyTargetConnection に移動して選択します。
 - b. **[編集]** をクリックします。**[値の編集]** ダイアログボックスが開きます。
 - c. **[ソース]** の横のフィールドで、**[コンテンツ]** を選択します。
 - d. **[値]** の横のフィールドで、**[マイファイル接続]** を選択します。

7. 次の手順を実行して、MyTargetObject を【データタスク 1】の【入力フィールド】セクションに追加します。
- 【追加】アイコンをクリックし、MyTargetObject に移動して選択します。
 - 【編集】をクリックします。【値の編集】ダイアログボックスが開きます。
 - 【ソース】の横のフィールドで、【コンテンツ】を選択します。
 - 【値】の横のフィールドで、**output10.txt** と入力します。
- 次の図は、データタスク 1 の【入力フィールド】を示しています。

</

ターゲットデータオブジェクトパラメータ MyTargetObject は、デフォルト値の **output5.txt** から、**output10.txt** に書き換えられています。

注: 書き換えるパラメータを【入力フィールド】セクションに追加するだけで済みます。この例では、MyTargetObject のみを書き換えます。ただし、MySourceObject、MySourceObject、および MyTargetConnection を書き換えるオプションもあります。

パラメータセット

パラメータセットとは、ユーザー定義パラメータとそれに関連する値のリストです。

パラメータセットを割り当てて、タスクフロー入力パラメータの値を指定し、タスクフローを実行できます。パラメータセットを使用すると、タスクフローを編集せずに、更新する値を定義できます。タスクフローはパラメータセットからパラメータを読み取り、タスクフローの実行時に値が適用されます。

パラメータセットの要件

タスクフロー間でパラメータセットを再利用できます。タスクフロー内のサブタスクフローステップとデータタスクステップのパラメータを定義することもできます。パラメータセットを再利用するには、パラメータセット内でローカルパラメータとグローバルパラメータを定義します。

パラメータセットの異なるセクションにあるパラメータをグループ化できます。各セクションの前には、パラメータ値を適用するプロジェクト、フォルダ、およびアセットを識別する見出しが付ききます。見出しのすぐ下でパラメータを定義します。このとき各パラメータを新しい行に入力します。

以下の表では、パラメータセットの各セクションを定義する見出し、および各セクションに定義したパラメータのスコープについて説明します。

見出し	説明
#USE_SECTIONS	この見出しを、セクションが含まれるパラメータセットの最初の行として使用します。そうでない場合、タスクフローは最初のグローバルセクションのみを読み取り、その他すべてのセクションを無視します。
[グローバル]	すべてのプロジェクト、フォルダ、タスクフロー、サブタスクフローステップ、およびデータタスクステップのパラメータを定義します。
[プロジェクト名].[フォルダ名].[タスクフロー名] または [プロジェクト名].[タスクフロー名]	特定のタスクフローのパラメータを定義します。 タスクフローセクションおよびグローバルセクションでパラメータが定義されている場合、タスクフローセクションの値でグローバル値が上書きされます。タスクフローセクションでパラメータ値が定義されていない場合は、グローバル値が考慮されます。
[プロジェクト名].[フォルダ名].[タスクフロー名].[サブタスクフローステップ名] または [プロジェクト名].[タスクフロー名].[サブタスクフローステップ名]	特定のサブタスクフローステップのパラメータを定義します。 サブタスクフローステップセクションおよびグローバルセクションでパラメータが定義されている場合、サブタスクフローステップセクションの値でグローバル値が上書きされます。 パラメータがサブタスクフローステップセクションとタスクフローセクションで定義され、タスクフローがサブタスクフローステップを使用する場合、サブタスクフローステップセクションの値が考慮されます。 サブタスクフローステップまたはタスクフローセクションでパラメータ値が定義されていない場合は、グローバル値が考慮されます。
[プロジェクト名].[フォルダ名].[タスクフロー名].[データタスクステップ名] または [プロジェクト名].[タスクフロー名].[データタスクステップ名] または [プロジェクト名].[フォルダ名].[タスクフロー名].[サブタスクフローステップ名]...[データタスクステップ名] または [プロジェクト名].[タスクフロー名].[サブタスクフローステップ名]...[データタスクステップ名]	特定のデータタスクステップのパラメータを定義します。 データタスクステップセクションおよびグローバルセクションでパラメータが定義されている場合、データタスクステップセクションの値でグローバル値が上書きされます。 パラメータがデータタスクステップセクション、タスクフロー、またはサブタスクフローステップセクションで定義され、タスクフローまたはサブタスクフローがデータタスクステップを使用する場合、データタスクステップセクションの値が考慮されます。 パラメータ値がデータタスクステップまたは直前のレベルのタスクフローまたはサブタスクフローステップセクションで定義されていない場合、グローバル値が考慮されます。
[プロジェクト名].[マッピングタスク名] または [プロジェクト名].[フォルダ名].[マッピングタスク名]	特定のマッピングタスクのパラメータを定義します。 マッピングタスクセクションおよびグローバルセクションでパラメータが定義されている場合、マッピングタスクセクションの値でグローバル値が上書きされます。

パラメータセットにセクションが含まれない場合、タスクフローはすべてのパラメータをグローバルとして読み取ります。

次のようにパラメータ名の前に 2 つのドル記号を付けます。

\$\$<parameter>

次のようにパラメータ値を定義します。

```
$$<parameter>=value1
```

```
$$<parameter2>=value2
```

パラメータ値には、先頭または末尾のスペースを含めて、等号 (=) の後のすべての文字が含まれます。パラメータ値は文字列値として扱われます。

パラメータスコープ

パラメータセットの複数のセクションで同じパラメータの値を定義した場合、スコープが最も小さいパラメータが、スコープが大きいパラメータよりも優先されます。

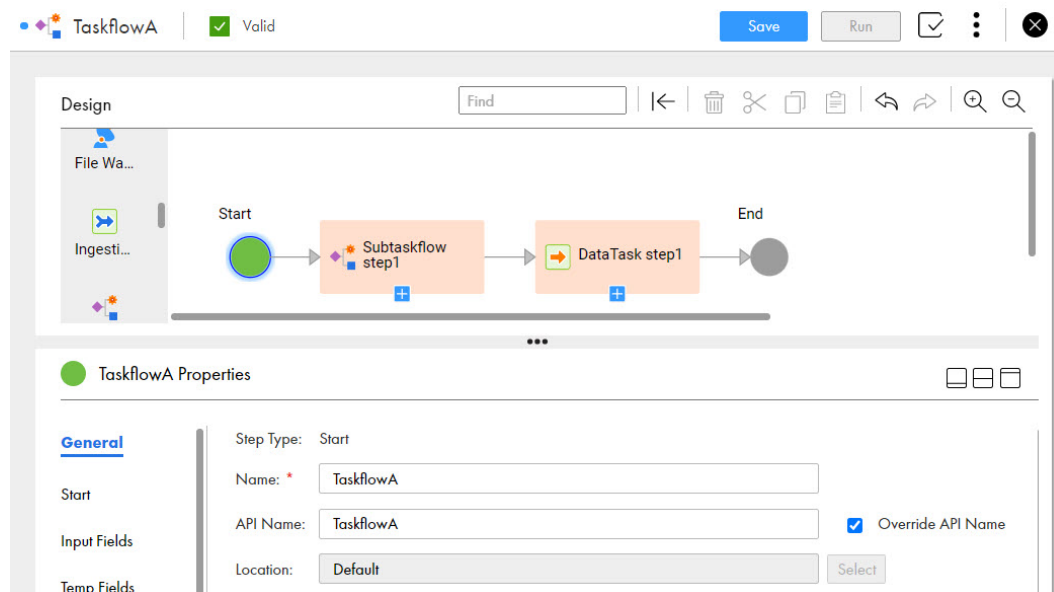
この場合、タスクフローは次のような順序でパラメータ値を優先します。

1. マッピングタスクセクションで定義した値。
2. データタスクステップセクションで定義した値。
3. サブタスクフローステップセクションで定義した値。
4. タスクフローセクションで定義した値。
5. #USE_SECTIONS で定義した値。
6. グローバルセクションで定義した値。

マッピングタスク、タスクフロー、サブタスクフローステップ、またはデータタスクステップセクションでパラメータを定義し、データタスクステップがマッピングタスクを使用する場合、タスクフローはマッピングタスクセクションで定義されたパラメータ値を使用します。

データタスクステップセクション、タスクフロー、またはサブタスクフローステップセクションでパラメータを定義し、タスクフローがデータタスクステップを使用する場合、タスクフローはデータタスクステップセクションで定義されたパラメータ値を使用します。同様に、サブタスクフローステップセクションとタスクフローセクションでパラメータを定義し、タスクフローがサブタスクフローステップを使用する場合、タスクフローはサブタスクフローステップセクションで定義されたパラメータ値を使用します。

例えば、次の図に示すように、Subtaskflow step1 という名前のサブタスクフローステップと、DataTask step1 という名前のデータタスクステップを含むタスクフローがあるとします。



次のパラメータ値をパラメータセットに定義します。

```
#USE_SECTIONS
$$source=customer_table
[GLOBAL]
$$location=USA
$$sourceconnection=Oracle
[Default].[Sales].[TaskflowA]
$$source=Leads_table
$$sourceconnection=Oracle_DB
$$sourcetype=Database
[Default].[Sales].[TaskflowA].[Subtaskflow step1]
$$source=Revenue
$$sourceconnection=ODBC
[Default].[Sales].[DataTask step1]
$$source=Account_table
```

DataTask step1 には、\$\$location、\$\$source、および\$\$sourceconnection パラメータが含まれています。

TaskflowA には、Subtaskflow step1 と DataTask step1 が含まれています。

TaskflowA を実行すると、TaskflowA は次のパラメータ値を使用します。

パラメータ	セクション	値
\$\$source	[Default].[Sales].[DataTask step1]	Account_table
\$\$sourceconnection	[Default].[Sales].[TaskflowA]	Oracle_DB
\$\$location	[GLOBAL]	USA

Subtaskflow step1 を実行すると、タスクフローは次のパラメータ値を使用します。

パラメータ	セクション	値
\$\$source	[Default].[Sales].[TaskflowA].[Subtaskflow step1]	Revenue
\$\$sourceconnection	[Default].[Sales].[TaskflowA].[Subtaskflow step1]	ODBC
\$\$location	[GLOBAL]	USA
\$\$sourcetype	[Default].[Sales].[TaskflowA]	Database

DataTask step1 を実行すると、タスクフローは次のパラメータ値を使用します。

パラメータ	セクション	値
\$\$source	[Default].[Sales].[DataTask step1]	Account_table
\$\$sourceconnection	[Default].[Sales].[TaskflowA]	Oracle_DB
\$\$location	[GLOBAL]	USA

\$\$source パラメータを含む他のすべてのアセットについて、タスクフローは値 customer_table を使用します。

パラメータセットのサンプル

次の例は、タスクフローのパラメータ値を含むサンプルのパラメータセットを示しています。

```
#USE_SECTIONS
$$source=customer_table
[GLOBAL]
$$sourceconnection=Oracle

[Project1].[Folder1].[Taskflow1]
$SourceConnection1=OracleStagingSourceConnection1
$SourceObject1=Source_stage1
$TargetConnection1=OracleStagingTargetConnection1
$TargetObject1=Target_stage1

$SourceConnection2=OracleStagingSourceConnection2
$SourceObject2=Source_stage2
$TargetConnection2=OracleStagingTargetConnection2
$TargetObject2=Target_stage2

$SourceConnection3=OracleProdSourceConnection
$SourceObject3=Source_prod
$TargetConnection3=OracleProdTargetConnection
$TargetObject3=Target_prod
```

タスクフローは、パラメータを使用して、ソース、ソース接続、ターゲット、およびターゲット接続に異なる値を渡します。

パラメータセットに関するルールおよびガイドライン

タスクフローでパラメータセットを使用する場合は、次のルールとガイドラインを考慮してください。

- タスクフローで使用するパラメータセットがクラウドホステッドリポジトリで使用できない場合、タスクフローは実行時に失敗します。パラメータセットをアップロードしてタスクフローを再度実行するか、実行時にパラメータセット名を有効なパラメータセット名で上書きできます。
- 渡す入力フィールドに、パラメータセットで指定された値がない場合、タスクフローはデータ型のデフォルト値を使用します。例えば、Boolean の場合、値は False、整数はゼロ、文字列は NULL のようになります。
- パラメータ値が、セットで定義されている別のパラメータの場合、タスクフローは最も限定されたスコープで変数の最初の値を使用します。例えば、パラメータセットに以下のようなパラメータ値が含まれていたとします。

```
[GLOBAL]
$$ffconnection=my_ff_conn
$$var2=California
$$var6=$var5
$var5=North
[Default].[folder5].[sales_accounts]
$$var2=$var5
$var5=south
```

「sales_accounts」で、「var5」の値は「south」です。var2 は var5 として定義されているので、var2 も「south」です。

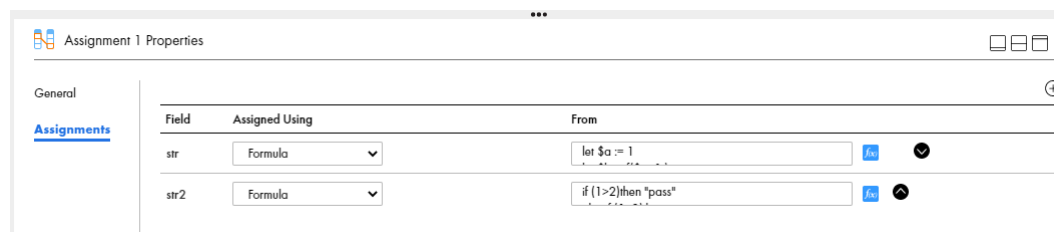
- セクション内に存在する場合を除き、パラメータの値はグローバルです。
- パラメータセットに同じ見出しが複数回含まれている場合、タスクフローは最初のセクションのパラメータを使用します。

式エディタ

式エディタで式を作成するには、フィールド、関数、および演算子を使用します。

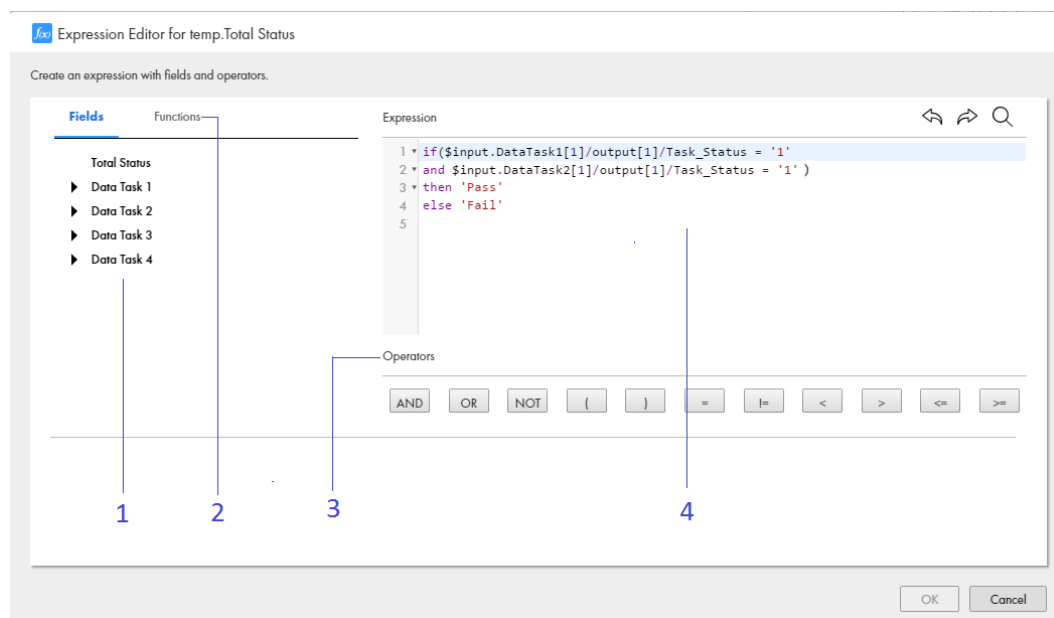
値【計算式】をフィールドに割り当てる場合は、データの取得元となるフィールドの計算式または式を作成する必要があります。式を作成するには、式エディタを使用します。

次の図では、フィールド【str】および【str2】に値【計算式】があります。



式エディタを開くには、値【計算式】フィールドの横にある **f(x)** をクリックします。

次の図は、【式エディタ】ダイアログボックスを示しています。



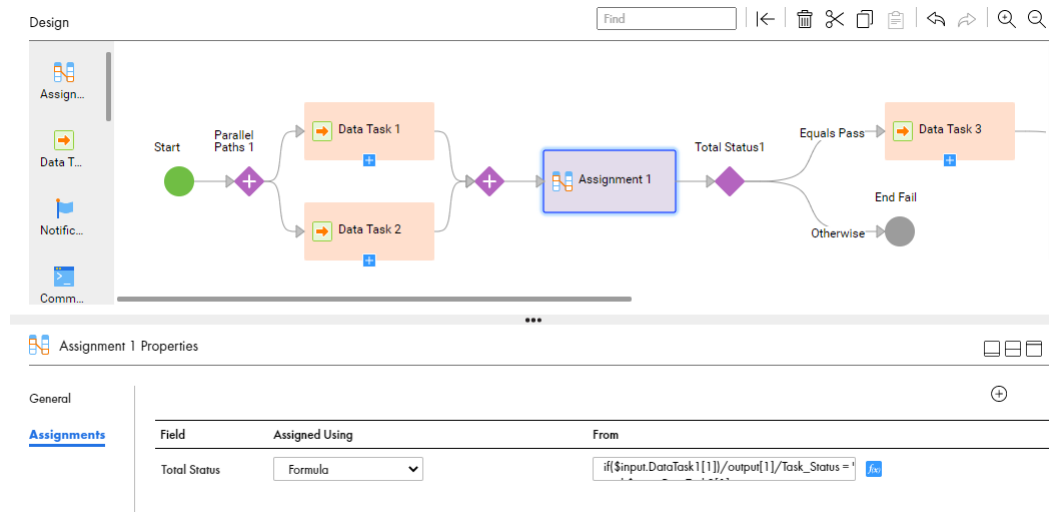
式エディタには、次のセクションがあります。

- セクション 1、【フィールド】セクション。定義した入力、出力、および一時フィールドの一覧がここに表示されます。
- セクション 2、【関数】セクション。ここでは、一般的な XQuery 関数の一覧が表示されます。意味を表示する関数を選択します。
- セクション 3、【演算子】セクション。式の作成に使用できる演算子の一覧がここに表示されます。
- セクション 4、【式】セクション。作成した式がここに表示されます。使用する条件と演算子では、大文字と小文字が区別されます。

図に示す式は、並行して実行される 3 つのタスクが成功した場合に、一時フィールド【合計ステータス】を Pass と定義します。

次に、タスクフローは、データ決定ステップで【合計ステータス】を使用します。【合計ステータス】の値が Pass である場合、タスクフローは別のデータタスクを実行します。【合計ステータス】の値が Fail である場合には、タスクフローは終了します。

次の図は、【合計ステータス】を使用するタスクフローを示しています。



式を作成するには、次のオプションを使用します。

- フィールドを追加するには、【フィールド】タブをクリックし、使用するフィールドにドリルダウンして、【追加】をクリックします。
- 演算子を追加するには、【演算子】セクションで演算子をクリックします。演算子を手動で入力することもできます。例えば、If 演算子を手動で入力します。
- 関数を追加するには、【関数】タブをクリックし、使用する関数をドリルダウンして【追加】をクリックします。
- コメントを追加するには、(<comment>)という構文【式】セクションにコメントを入力します。例えば、(:This is a sample comment:)と入力します。

コメントを使用すると、式に関する説明を付記できます。または、式に関連するビジネスドキュメントにアクセスする URL を指定できます。

式エディタは、入力した式を検証します。無効な式は保存できません。

ヒント: XQuery 3.0 を使用した式の作成

XQuery バージョン 3.0 を使用して、式エディタで式を作成します。このトピックの例では、単一のステートメントと複数のステートメントの XQuery 式を作成するために使用する、構文と要素について示します。

XQuery 3.0 の詳細については、<https://www.w3.org/TR/xquery-30/>を参照してください。

単一ステートメントの式

次の式は、単一のステートメント式です。

```
concat("Hello", " ", $input.n1)
```

以下の注記では、この式の各部分を説明しています。

- concat は、2 つ以上の値を 1 つの文字列に結合する関数です。

- "hello"、" " \$input.n1 は、関数 concat のパラメータです。
 - "hello"は文字列です。常に引用符で文字列を囲みます。一重引用符または二重引用符のどちらを使用してもかまいません。ただし、式内で同じスタイルを使用するようにしてください。
 - n1 は入力変数です。式に追加すると、式エディタによって\$input.n1 に変換されます。変数の先頭には、常に\$を付けます。変数の周囲には引用符を追加しないでください。
 - パラメータ" "はスペースを表します。
- パラメータはかっこで囲みます。
- パラメータはコンマで区切ります。

\$n1 の値が「World」であるとします。

以下を実行するとします。

```
concat("Hello", " ", $input.n1)
```

出力は以下のようになります。

Hello World

マルチステートメント式

次の式は、マルチステートメント式です。

```
let $n1 := number($input.n1)
let $n2 := number($input.n2)

let $r1 := if ($n1 > $n2)
            then "Greater: N1 > N2"
            else if ($n1 < $n2)
            then "Less: N1 < N2"
            else "Same"

return $r1
```

以下の注記では、この式の各部分を説明しています。

- 最初に、変数\$n1 と \$n2 を宣言します。演算子:=を使用して、値を割り当てます。

注: \$n1 および \$n2 を割り当てステップで整数として定義した場合でも、式エディタで数値として宣言する必要があります。
- let、if、then、および else などの XQuery キーワードを使用できます。これらは大文字小文字を区別します。

重要: let をキーワードとして式を開始する場合は、キーワード return を使用して式を終了する必要があります。
- 次のルールを使用して、\$r1 という 3 番目の変数の値を定義するには、式エディタを使用します。
 - \$n1 の値が \$n2 の値よりも大きい場合、\$r1 は値 Greater: N1 > N2 を受け取ります。
 - \$n1 の値が \$n2 の値よりも小さい場合、\$r1 は値 Less: N1 < N2 を受け取ります。
 - \$n1 の値と \$n2 の値が同じである場合、\$r1 は値 Same を受け取ります。

\$n1 の値が 20 であり、\$n2 の値が 250 であると想定します。

以下を実行するとします。

```
let $n1 := number($input.n1)
let $n2 := number($input.n2)

let $r1 := if ($n1 > $n2)
            then "Greater: N1 > N2"
            else if ($n1 < $n2)
            then "Less: N1 < N2"
```



```
else "Same"

return $r1
```

出力は以下のようになります。

Less: N1<N2

ここで、\$r1 は次の値を持ちます。

Less: N1<N2

キーボードショートカット

式を作成するときに、キーボードショートカットを使用できます。

キーボードショートカットを使用するには、[式] セクション内にポインタを置きます。

以下のキーボードショートカットを利用できます。

アクション	ショートカット
元に戻す	Ctrl+Z
やり直し	Ctrl+Y
コピー	Ctrl+C
切り取り	Ctrl+X
貼り付け	Ctrl+V
検索	Ctrl+F
4 スペース分をインデント	Tab
使用可能な変数の一覧を表示	\$
コード補完、つまり、使用可能な挿入の一覧を表示します。挿入には、名前空間、関数、フィールド、または一般的なコードフラグメントがあります。	Ctrl+スペース

タスクフロー関数

タスクフロー式エディタでは複数の関数を使用できます。

主な関数の一部を以下に示します。

アセット詳細機能

式エディタの **【その他】** セクションから、次のアセット詳細関数を使用できます。

- getAssetLocation
- getAssetName
- getInstanceStartTime

文字関数

式エディタの【文字列】セクションから、次の文字関数を使用できます。

- instr
- lpad
- ltrim
- rtrim

変換関数

式エディタの【日付と時刻】、【その他】、または【文字列】セクションから、次の変換関数を使用できます。

- toChar(Numbers)
- toDate
- toDecimal
- toInteger

データクレンジング関数

式エディタの【その他】セクションから、次のデータクレンジング関数を使用できます。

- in

日付関数

式エディタの【日付と時刻】セクションから、次の日付関数を使用できます。

- addToDate
- dateDiff
- getDatePart
- lastDay
- trunc

数値関数

式エディタの【数値】セクションから、次の数値関数を使用できます。

- round(Numbers)

テスト関数

式エディタの【その他】セクションから、次のテスト関数を使用できます。

- decode
- iif
- isNull

addToDate

指定された量を Date/Time 値の一部に加算し、関数に渡した日付と同じ形式の日付を返します。addToDate 関数は、正と負の整数値を受け入れます。addToDate を使用して、日付の次の部分を変更します。

- 年。amount 引数に正または負の整数を入力します。年のフォーマット文字列として、Y、YY、YYY、YYYY のいずれかを使用できます。次の式は、SHIP_DATE カラムのすべての日付に 10 年を加えます。

```
date:addToDate(xs.dateTime('SHIP_DATE'), 'YY', 10)
```

- **月。***amount* 引数に正または負の整数を入力します。月のフォーマット文字列として、MM、MON、MONTH のいずれかを使用できます。次の式は SHIP_DATE カラムのすべての日付から 10 ヶ月を引きます。
date:addToDate(xs:dateTime('SHIP_DATE'), 'MONTH', -10)
- **日。***amount* 引数に正または負の整数を入力します。日のフォーマット文字列として、D、DD、DDD、DY、DAY のいずれかを使用できます。例えば、式は SHIP_DATE カラムのすべての日付に 10 日を加えます。
date:addToDate(xs:dateTime('SHIP_DATE'), 'DD', 10)
- **時間。***amount* 引数に正または負の整数を入力します。時間のフォーマット文字列として、HH、HH12、HH24 のいずれかを使用できます。以下の式は、SHIP_DATE カラムのすべての日付に 14 時間を加えます。
date:addToDate(xs:dateTime('SHIP_DATE'), 'HH', 14)
- **分。***amount* 引数に正または負の整数を入力します。分の設定にはフォーマット文字列 MI を使用します。以下の式は、SHIP_DATE カラムのすべての日付に 25 分を加えます。
date:addToDate(xs:dateTime('SHIP_DATE'), 'MI', 25)
- **秒。***amount* 引数に正または負の整数を入力します。秒の設定には SS フォーマット文字列を使用します。以下の式は、SHIP_DATE カラムのすべての日付に 59 秒を加えます。
date:addToDate(xs:dateTime('SHIP_DATE'), 'SS', 59)
- **ミリ秒。***amount* 引数に正または負の整数を入力します。秒の設定には MS フォーマット文字列を使用します。以下の式は、SHIP_DATE カラムのすべての日付に 125 ミリ秒を加えます。
date:addToDate(xs:dateTime('SHIP_DATE'), 'MS', 125)
- **マイクロ秒。***amount* 引数に正または負の整数を入力します。秒の設定には US フォーマット文字列を使用します。以下の式は、SHIP_DATE カラムのすべての日付に 2,000 マイクロ秒を加えます。
date:addToDate(xs:dateTime('SHIP_DATE'), 'US', 2000)

構文

date:addToDate(xs:dateTime('date'), 'format', amount)

注: xs:dateTime 句を手動で追加し、日付値を一重引用符で囲む必要があります。

次の表に、引数を示します。

引数	必須/オプション	説明
<i>日付</i>	必須	Date/Time データ型。変更を行う値を渡します。 有効なトランスフォーメーション式を必要に応じて入力できます。
<i>format</i>	必須	日付値の中の変更を行う部分を指定するフォーマット文字列。フォーマット文字列は 'mm' のように一重引用符で囲みます。フォーマット文字列は大文字と小文字を区別しません。
<i>amount</i>	必須	日付値を変更する値として、年数、月数、日数、時間数などを指定する整数値。整数を求める有効なトランスフォーメーション式を必要に応じて入力できます。

戻り値

関数に渡した日付と同じ形式の日付。

関数に引数として NULL 値が渡された場合には、NULL です。

例

以下の式はすべて、DATE_SHIPPED カラムの各日付に 1 か月を加算します。特定の月に存在しない日を作成する値を渡すと、addToDate はその月の最後の日を返します。例えば、Jan 31 1998 に 1 ヶ月を加えると、addToDate は Feb 28 1998 を返します。

また、addToDate はうるう年を認識し、Jan 29 2000 に 1 ヶ月を加えます。

```
date:addToDate(xs:dateTime('DATE_SHIPPED'), 'MM', 1)
date:addToDate(xs:dateTime('DATE_SHIPPED'), 'MON', 1)
date:addToDate(xs:dateTime('DATE_SHIPPED'), 'MONTH', 1)
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_SHIPPED	RETURN VALUE
Jan 12 1998 12:00:30AM	Feb 12 1998 12:00:30AM
Jan 31 1998 6:24:45PM	Feb 28 1998 6:24:45PM
Jan 29 2000 5:32:12AM	Feb 29 2000 5:32:12AM (<i>Leap Year</i>)
Oct 9 1998 2:30:12PM	Nov 9 1998 2:30:12PM
NULL	NULL

以下の式はすべて、DATE_SHIPPED カラムの各日付から 10 日を減算します。

```
date:addToDate(xs:dateTime('DATE_SHIPPED'), 'D', -10)
date:addToDate(xs:dateTime('DATE_SHIPPED'), 'DD', -10)
date:addToDate(xs:dateTime('DATE_SHIPPED'), 'DDD', -10)
date:addToDate(xs:dateTime('DATE_SHIPPED'), 'DY', -10)
date:addToDate(xs:dateTime('DATE_SHIPPED'), 'DAY', -10)
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:30AM	Dec 22 1996 12:00AM
Jan 31 1997 6:24:45PM	Jan 21 1997 6:24:45PM
Mar 9 1996 5:32:12AM	Feb 29 1996 5:32:12AM (<i>Leap Year</i>)
Oct 9 1997 2:30:12PM	Sep 30 1997 2:30:12PM
Mar 3 1996 5:12:20AM	Feb 22 1996 5:12:20AM
NULL	NULL

以下の式はすべて、DATE_SHIPPED カラムの各日付から 15 時間を減算します。

```
date:addToDate(xs:dateTime('DATE_SHIPPED'), 'HH', -15)
date:addToDate(xs:dateTime('DATE_SHIPPED'), 'HH12', -15)
date:addToDate(xs:dateTime('DATE_SHIPPED'), 'HH24', -15)
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:30AM	Dec 31 1996 9:00:30AM
Jan 31 1997 6:24:45PM	Jan 31 1997 3:24:45AM
Oct 9 1997 2:30:12PM	Oct 8 1997 11:30:12PM

DATE_SHIPPED	RETURN VALUE
Mar 3 1996 5:12:20AM	Mar 2 1996 2:12:20PM
Mar 1 1996 5:32:12AM	Feb 29 1996 2:32:12PM (<i>Leap Year</i>)
NULL	NULL

base64Decode

charSet 引数で指定された文字セットに基づいて、指定された入力文字列の Base64 デコードされたバージョンを返します。この関数は通常、添付ファイルに使用されます。

構文

```
util:base64Decode(data, charSet)
```

次の表に、引数を示します。

引数	必須/ オプション	説明
<i>data</i>	必須	String データ型。デコードするデータ。
<i>charSet</i>	オプション	データの文字デコード。タスクフローは、Azul JDK でエンコード用にサポートされる文字セットをサポートします。これには、US-ASCII、ISO-8859-1、UTF-8、UTF-16BE、UTF-16LE、および UTF-16 などが含まれます。 デフォルトは UTF-8 です。

戻り値

デコードされた値。

NULL 値を入力した場合は、NULL です。

例

MQSeries メッセージ ID をエンコードしてフラットファイルに書き込みました。フラットファイルソースから、MQSeries メッセージ ID を含むデータを読み込む必要があります。この場合、base64Decode を使用して ID をデコードし、元の文字列値に変換できます。

dateDiff

2 つの日付の間の時間の長さを返します。形式は、年、月、日、時間、分、秒、ミリ秒、またはマイクロ秒に指定できます。dateDiff 関数は、最初の日付から 2 番目の日付を減算し、差を返します。

dateDiff 関数は、日数ではなく月数に基づいて値を計算します。各月の選択された日で、1 カ月に満たない月の日付の差を計算します。1 ヶ月に満たない月の日付の差を計算するために、dateDiff は月内の日を加算します。次に、選択された月の全日数でその値を除算します。

dateDiff 関数は、うるう年とうるう年以外の年では同じ期間で異なる値を算出します。違いが発生するのは、dateDiff 関数に 2 月が含まれる場合です。dateDiff 関数は、うるう年の 2 月は 29 日間で除算し、うるう年以外の年である場合は 28 日間で除算します。

例えば、9 月 13 日から 2 月 19 日までの月数を計算するとします。うるう年の場合、dateDiff 関数は 2 月を 19/29 ヶ月、つまり 0.655 ヶ月と計算します。うるう年以外の年の場合、dateDiff 関数は 2 月を 19/28 ヶ月、

つまり 0.678 ヶ月と計算します。dateDiff 関数は、残りの月の日の差を同様に計算し、dateDiff 関数が指定された期間における合計値を返します。

注: 日付の差を計算するのに異なるアルゴリズムを使用するデータベースもあります。

構文

```
date:dateDiff(xs:dateTime('date'), xs:dateTime('date'), 'format')
```

注: xs:dateTime 句を手動で追加し、日付値を一重引用符で囲む必要があります。

次の表に、引数を示します。

引数	必須/オプション	説明
日付	必須	Date/Time データ型。比較を行う 1 つ目の日付の値を渡します。 有効なトランスフォーメーション式を必要に応じて入力できます。
日付	必須	Date/Time データ型。比較を行う 2 つ目の日付の値を渡します。 有効なトランスフォーメーション式を必要に応じて入力できます。
format	必須	日付または時間計測の単位を指定するフォーマット文字列。年、月、日、時間、分、秒、ミリ秒、またはマイクロ秒を指定できます。'mm' など、日付の中の 1 つの部分のみを指定できます。フォーマット文字列は一重引用符で囲んでください。フォーマット文字列は大文字と小文字を区別しません。 (例えば、フォーマット文字列 'mm' は 'MM'、'Mm'、'mM' と同じです。)

戻り値

Double 値。最初の日付が 2 番目の日付より後の場合、戻り値は正の数になります。最初の日付が 2 番目の日付より前の場合、戻り値は負の数になります。

2 つの日付が同じである場合は、ゼロです。

一方（または両方）の日付が NULL である場合は、NULL です。

例

次の式は、DATE_PROMISED カラムと DATE_SHIPPED カラムの日付の差が何時間あるかを返します。

```
date:dateDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'HH')  
date:dateDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'HH12')  
date:dateDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'HH24')
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:00AM	Mar 29 1997 12:00:00PM	-2100
Mar 29 1997 12:00:00PM	Jan 1 1997 12:00:00AM	2100
NULL	Dec 10 1997 5:55:10PM	NULL
Dec 10 1997 5:55:10PM	NULL	NULL
Jun 3 1997 1:13:46PM	Aug 23 1996 4:20:16PM	6812.89166666667
Feb 19 2004 12:00:00PM	Feb 19 2005 12:00:00PM	-8784

次の式は、DATE_PROMISED カラムと DATE_SHIPPED カラムの日付の差を日数で返します。

```
date:dateTimeDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'D')
date:dateTimeDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'DD')
date:dateTimeDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'DDD')
date:dateTimeDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'DY')
date:dateTimeDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'DAY')
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:00AM	Mar 29 1997 12:00:00PM	-87.5
Mar 29 1997 12:00:00PM	Jan 1 1997 12:00:00AM	87.5
NULL	Dec 10 1997 5:55:10PM	NULL
Dec 10 1997 5:55:10PM	NULL	NULL
Jun 3 1997 1:13:46PM	Aug 23 1996 4:20:16PM	283.870486111111
Feb 19 2004 12:00:00PM	Feb 19 2005 12:00:00PM	-366

次の式は、DATE_PROMISED カラムと DATE_SHIPPED カラムの日付の差を月数で返します。

```
date:dateTimeDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'MM')
date:dateTimeDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'MON')
date:dateTimeDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'MONTH')
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:00AM	Mar 29 1997 12:00:00PM	-2.91935483870968
Mar 29 1997 12:00:00PM	Jan 1 1997 12:00:00AM	2.91935483870968
NULL	Dec 10 1997 5:55:10PM	NULL
Dec 10 1997 5:55:10PM	NULL	NULL
Jun 3 1997 1:13:46PM	Aug 23 1996 4:20:16PM	9.3290162037037
Feb 19 2004 12:00:00PM	Feb 19 2005 12:00:00PM	-12

次の式は、DATE_PROMISED カラムと DATE_SHIPPED カラムの日付の差を年数で返します。

```
date:dateTimeDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'Y')
date:dateTimeDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'YY')
date:dateTimeDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'YYY')
date:dateTimeDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'YYYY')
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:00AM	Mar 29 1997 12:00:00PM	-0.24327956989247
Mar 29 1997 12:00:00PM	Jan 1 1997 12:00:00AM	0.24327956989247

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
NULL	Dec 10 1997 5:55:10PM	NULL
Dec 10 1997 5:55:10PM	NULL	NULL
Jun 3 1997 1:13:46PM	Aug 23 1996 4:20:16PM	0.77741801697531
Feb 19 2004 12:00:00PM	Feb 19 2005 12:00:00PM	-1

次の式は、DATE_PROMISED カラムと DATE_SHIPPED カラムの日付の差を月数で返します。

```
date:dateTimeDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'MM')
date:dateTimeDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'MON')
date:dateTimeDiff(xs:dateTime('DATE_PROMISED'), xs:dateTime('DATE_SHIPPED'), 'MONTH')
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_PROMISED	DATE_SHIPPED	LEAP YEAR VALUE (in Months)	NON-LEAP YEAR VALUE (in Months)
Sept 13	Feb 19	-5.237931034	-5.260714286
NULL	Feb 19	NULL	N/A
Sept 13	NULL	NULL	N/A

decode

指定した値をカラムで検索します。値を見つけると、定義された結果値を返します。1 つの decode 関数内には、数に制限なく検索を指定できます。

decode を使って文字列カラムの値を検索する場合には、 rtrim 関数で末尾の空白を削除することや、検索文字列内に空白文字を含めることもできます。

構文

```
util:decode(value, search1, result1, args, default)
```


次の表に、引数を示します。

引数	必須/ オプション	説明
<i>value</i>	必須	検索を行う値を渡します。 任意の有効なトランスフォーメーション式を入力できます。バイナリ以外の任意のデータ型を渡すことができます。 NULL 値を渡すには、次の形式で空のシーケンスを指定する必要があります: ()
<i>search1</i>	必須	検索を行う対象の値を渡します。 任意の有効なトランスフォーメーション式を入力できます。value 引数と同じデータ型の任意の値を渡すことができます。検索する値と値引数は、必ず一致させる必要があります。値の一部を検索することはできません。また、検索する値では大文字と小文字が区別されます。 例えば、特定のカラムで文字列'Halogen Flashlight'を検索したい場合には、'Halogen'だけでなく'Halogen Flashlight'と入力する必要があります。'Halogen'と入力すると、一致する値が見つかりません。 NULL 値を渡すには、次の形式で空のシーケンスを指定する必要があります: ()
<i>result1</i>	必須	検索で一致した値が見つかったときに返す値です。 任意の有効なトランスフォーメーション式を入力して、バイナリ以外の任意のデータ型を渡すことができます。 NULL 値を渡すには、次の形式で空のシーケンスを指定する必要があります: ()
<i>args</i>	必須	カンマで区切られた検索値と結果値のペア。 例えば、以下の構文を使用します。 util:decode(value, search1, result1, search2, result2, searchn, resultn, default) NULL 値を渡すには、次の形式で空のシーケンスを指定する必要があります: ()
<i>default</i>	必須	検索で一致した値が見つからなかったときに返す値です。 任意の有効なトランスフォーメーション式を入力して、バイナリ以外の任意のデータ型を渡すことができます。 NULL 値を渡すには、次の形式で空のシーケンスを指定する必要があります: ()

戻り値

一致した値が見つかった場合は、*result1*。

一致した値が見つからなかった場合は、*default* 値。

一致した値が見つからなかった場合、デフォルト値を指定していなければ NULL。

複数の条件に一致した場合でも、decode は最初に一致した結果を返します。

decode とデータ型

decode を使用した場合、戻り値のデータ型は最大の精度を持つ結果のデータ型と常に同じものとなります。

たとえば、次のような式があるとします。

```
util:decode( CONST_NAME
    'Five', 5,
    'Pythagoras', 1.414213562,
    'Archimedes', 3.141592654,
    'Pi', 3.141592654 )
```

この式の戻り値は、5、1.414213562、および 3.141592654 です。最初の結果は整数で、その他の結果は小数です。Decimal データ型は、Integer データ型よりも精度が高くなります。この式は常に結果を固定小数点値として書き込みます。

文字列戻り値と数値戻り値の両方を持つ DECODE 関数は作成できません。

例えば、下記の式は有効ではありません。

```
util:decode( CONST_NAME
             'Five', 5,
             'Pythagoras', '1.414213562',
             'Archimedes', '3.141592654',
             'Pi', 3.141592654 )
```

上記の式を検査すると、下記のエラーメッセージが表示されます。

Function cannot resolve operands of ambiguously mismatching datatypes.

例

次の式は、decode を使用し、特定の ITEM_ID を検索して ITEM_NAME を返します。

```
util:decode( ITEM_ID, 10, 'Flashlight',
             14, 'Regulator',
             20, 'Knife',
             40, 'Tank',
             'NONE' )
```

次の表に、一部のサンプル値と戻り値を示します。

ITEM_ID	RETURN VALUE
10	Flashlight
14	Regulator
17	NONE
20	Knife
25	NONE
NULL	NONE
40	Tank

検索値が ITEM_ID に一致しないため、decode は項目 17 および 25 に対してデフォルト値の NONE を返します。また、NULL の ITEM_ID に対しても NONE を返します。

次の式は、複数のカラムおよび条件をテストして、上から順に'TRUE'であるか'FALSE'であるかを求めます。

```
util:decode( TRUE,
             Var1 = 22, 'Variable 1 was 22!',
             Var2 = 49, 'Variable 2 was 49!',
             Var1 < 23, 'Variable 1 was less than 23.',
             Var2 > 30, 'Variable 2 was more than 30.',
             'Variables were out of desired ranges.')
```

次の表に、一部のサンプル値と戻り値を示します。

Var1	Var2	RETURN VALUE
21	47	Variable 1 was less than 23.

Var1	Var2	RETURN VALUE
22	49	Variable 1 was 22!
23	49	Variable 2 was 49!
24	27	Variables were out of desired ranges.
25	50	Variable 2 was more than 30.

getAssetLocation

関数を使用するタスクフローが保存されている場所を返します。

例えば、この関数を使用して、デバッグを目的としてタスクフローの場所を見つけることができます。

構文

```
util:getAssetLocation()
```

getAssetLocation 関数は引数を使用しません。

戻り値

関数を使用するタスクフローが保存される場所。

例

Default\Orders の下に格納されているタスクフローで getAssetLocation 関数を使用すると、関数は次の値を返します。

```
Default\Orders
```

getAssetName

関数を使用するタスクフローの名前を返します。

例えば、通知タスクステップの関数を使用して、ステークホルダーに送信するタスクフローの失敗の電子メール通知にタスクフロー名を含めることができます。

構文

```
util:getAssetName()
```

getAssetName 関数は引数を使用しません。

戻り値

関数を使用するタスクフローの名前。

例

Order Management という名前のタスクフローで getAssetName 関数を使用すると、関数は次の値を返します。

```
Order Management
```

getDatePart

日付の中の指定した部分を整数値として返します。そのため、日付の月の部分を返す式を作成して「Apr 1 1997 00:00:00」のような日付を渡すと、getDatePart は 4 を返します。

構文

```
date:getDatePart(xs:dateTime('date'), 'format')
```

注: xs:dateTime 句を手動で追加し、日付値を一重引用符で囲む必要があります。

次の表に、引数を示します。

引数	必須/オプション	説明
<i>date</i>	必須	Date/Time データ型。 有効なトランスフォーメーション式を必要に応じて入力できます。
<i>format</i>	必須	日付値の中で返す部分を指定するフォーマット文字列。フォーマット文字列は'mm'のように一重引用符で囲みます。フォーマット文字列は大文字と小文字を区別しません。 例えば、日付「Apr 1 1997」を getDatePart に渡す場合、フォーマット文字列 'Y'、'YY'、'YYY'、'YYYY' はすべて 1997 を返します。

戻り値

日付の中の指定された部分を示す整数。

関数に NULL 値を渡した場合は NULL です。

例

以下の式は、DATE_SHIPPED カラムの各日付の時の部分を返します。デフォルトの日付形式は 24 時間方式に基づくため、12:00:00AM は 0 を返します。

```
date:getDatePart(xs:dateTime('DATE_SHIPPED'), 'HH')
date:getDatePart(xs:dateTime('DATE_SHIPPED'), 'HH12')
date:getDatePart(xs:dateTime('DATE_SHIPPED'), 'HH24')
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	0
Sep 2 1997 2:00:01AM	2
Aug 22 1997 12:00:00PM	12
June 3 1997 11:30:44PM	23
NULL	NULL

以下の式は、DATE_SHIPPED カラムの各日付の日の部分を返します。

```
date:getDatePart(xs:dateTime('DATE_SHIPPED'), 'D')
date:getDatePart(xs:dateTime('DATE_SHIPPED'), 'DD')
date:getDatePart(xs:dateTime('DATE_SHIPPED'), 'DDD')
date:getDatePart(xs:dateTime('DATE_SHIPPED'), 'DY')
date:getDatePart(xs:dateTime('DATE_SHIPPED'), 'DAY')
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	13
June 3 1997 11:30:44PM	3
Aug 22 1997 12:00:00PM	22
NULL	NULL

以下の式は、DATE_SHIPPED カラムの各日付の月の部分を返します。

```
date.getDatePart(xs:dateTime('DATE_SHIPPED'), 'MM')  
date.getDatePart(xs:dateTime('DATE_SHIPPED'), 'MON')  
date.getDatePart(xs:dateTime('DATE_SHIPPED'), 'MONTH')
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	3
June 3 1997 11:30:44PM	6
NULL	NULL

以下の式は、DATE_SHIPPED カラムの各日付の年の部分を返します。

```
date.getDatePart(xs:dateTime('DATE_SHIPPED'), 'Y')  
date.getDatePart(xs:dateTime('DATE_SHIPPED'), 'YY')  
date.getDatePart(xs:dateTime('DATE_SHIPPED'), 'YYY')  
date.getDatePart(xs:dateTime('DATE_SHIPPED'), 'YYYY')
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	1997
June 3 1997 11:30:44PM	1997
NULL	NULL

getInstanceStartTime

関数を使用するタスクフローの実行中のインスタンスの開始日と開始時刻を返します。

例えば、この関数を使用して、タスクフローの実行がいつ開始されたかを確認し、予想される期間を超えて実行されている場合はタスクフローを強制終了できます。

構文

```
util.getInstanceStartTime()
```

getInstanceStartTime 関数は引数を使用しません。

戻り値

関数を使用するタスクフローの実行中のインスタンスの開始日と開始時刻。

戻り値は、次のような協定世界時（UTC）形式になります。

YYY-MM-DDTHH:mm:ss.sssZ

例

2021 年 1 月 19 日に開始されたタスクフローで getInstanceStartTime 関数を使用すると、関数は次の値を返します。

2021-01-19T10:11:21.047Z

iif

条件の結果に基づいて、指定した 2 つの値のうちの 1 つを返します。

構文

util:iif(*condition*, *val1*, *val2*)

次の表に、引数を示します。

引数	必須/オプション	説明
<i>condition</i>	必須	評価を行う条件。 TRUE または FALSE になる有効なトランスフォーメーション式を必要に応じて入力できます。
<i>val1</i>	必須	条件が TRUE のときに返す値。戻り値は常にこの引数で指定したデータ型になります。 別の iif 式を含む、任意の有効なトランスフォーメーション式を入力できます。バイナリ以外の任意のデータ型を渡すことができます。
<i>val2</i>	オプション	条件が FALSE のときに返す値。 別の iif 式を含む、任意の有効なトランスフォーメーション式を入力できます。バイナリ以外の任意のデータ型を渡すことができます。

iif 関数に FALSE (*val2*) 条件は必要ありません。*val2* を省略すると、条件が FALSE の場合、関数は次のいずれかの値を返します。

- *val1* が Numeric データ型の場合は 0。
- *val1* が String データ型の場合は、空の文字列。
- *val1* が Date/Time データ型の場合は NULL。

例えば、次の式には FALSE 条件が含まれておらず、*val1* は String データ型であるため、decode は FALSE と評価されたそれぞれの行に対して空の文字列を返します。

util:iif(SALES > 100, EMP_NAME)

次の表に、一部のサンプル値と戻り値を示します。

SALES	EMP_NAME	RETURN VALUE
150	John Smith	John Smith

SALES	EMP_NAME	RETURN VALUE
50	Pierre Bleu	' ' (<i>empty string</i>)
120	Sally Green	Sally Green
NULL	Greg Jones	' ' (<i>empty string</i>)

戻り値

条件が TRUE の場合は *val1*。

条件が FALSE の場合は *val2*。

例えば、下記の式には FALSE 条件である NULL が含まれるため、decode は FALSE と評価されたそれぞれの行に対して NULL を返します。

```
util:iif(SALES > 100, EMP_NAME, NULL)
```

次の表に、一部のサンプル値と戻り値を示します。

SALES	EMP_NAME	RETURN VALUE
150	John Smith	John Smith
50	Pierre Bleu	NULL
120	Sally Green	Sally Green
NULL	Greg Jones	NULL

iif およびデータ型

iif を使用した場合、戻り値のデータ型は最大の精度を持つ結果のデータ型と同じものとなります。

例えば、次のような式があるとします。

```
util:iif(SALES < 100, 1, .3333)
```

TRUE の結果 (1) は整数であり、FALSE の結果 (.3333) は小数です。Decimal データ型は、Integer データ型よりも精度が高くなります。したがって、戻り値のデータ型は常に固定小数点値です。

IIF の特殊な使用法

ネストした IIF 文を使用して、複数の条件をテストできます。以下の例は、各種条件をテストし、販売額が 0 または負の場合には 0 を返します。

```
util:iif(SALES > 0, util:iif(SALES < 50, SALARY1, util:iif(SALES < 100, SALARY2, util:iif(SALES < 200, SALARY3, BONUS))), 0 )
```

in

入力値を値のリストとマッチングします。デフォルトでは、大文字と小文字を区別します。

構文

```
util:in(valueToSearch, values, caseFlag)
```

次の表に、引数を示します。

引数	必須/ オプション	説明
valueToSearch	必須	文字列、日付または数値。値のカンマ区切りリストと照合する入力値。 NULL 値を渡すには、次の形式で空のシーケンスを指定する必要があります: ()
values	必須	文字列、日付または数値。検索する値のカンマ区切りリスト。値にはカラムを使用できます。指定する値の数に制限はありません。 NULL 値を渡すには、次の形式で空のシーケンスを指定する必要があります: ()
caseFlag	オプション	整数でなければなりません。この関数の引数の大文字と小文字を区別するかどうかを指定します。有効なトランスフォーメーション式を必要に応じて入力できます。 <i>caseFlag</i> が 0 以外の数値の場合、大文字と小文字を区別します。 <i>caseFlag</i> が NULL 値または 0 の場合、大文字と小文字を区別しません。

戻り値

入力値が値リストに一致する場合は TRUE (1)。

入力値が値リストに一致しない場合は FALSE (0)。

NULL 値を入力した場合は、NULL です。

例

以下の式は、入力値が safety Knife、Chisel Point Knife、Medium Titanium Knife のいずれであるかを決定します。入力値は、必ずしもカンマ区切りリストの値の表記（大文字/小文字）に一致させる必要はありません。

```
util:in(ITEM_NAME, 'Chisel Point Knife', 'Medium Titanium Knife', 'Safety Knife', 0)
```

次の表に、一部のサンプル値と戻り値を示します。

ITEM_NAME	RETURN VALUE
Stabilizing Vest	0 (FALSE)
Safety knife	1 (TRUE)
Medium Titanium knife	1 (TRUE)
	NULL

instr

文字列の中で、指定した文字が左から数えて何文字目にあるかを返します。

構文

```
sff:instr(str, search, start, occurrence, comparison_type)
```


次の表に、引数を示します。

引数	必須/ オプション	説明
<i>str</i>	必須	評価を行う値を渡します。 文字列はすべて文字である必要があります。有効なトランスフォーメーション式を必要に応じて入力できます。 式の結果は文字列である必要があります。文字列ではない場合、評価の前に <i>instr</i> によって値が文字列に変換されます。
<i>search</i>	必須	検索を行う文字または文字列を指定します。 有効なトランスフォーメーション式を必要に応じて入力できます。文字列を検索する場合は、検索する文字を一重引用符または二重引用符で囲みます。 値は、文字列の一部に一致する必要があります。例えば、 <code>INSTR('Alfred Pope', 'Alfred Smith')</code> を使用した場合、関数は 0 を返します。 値は大文字と小文字が区別されます。
<i>start</i>	オプション	文字列内で検索を開始する位置を示します。有効なトランスフォーメーション式を必要に応じて入力できます。値は整数でなければなりません。 デフォルト値は 1 です。つまり、 <i>instr</i> は文字列の最初の文字から検索を開始します。 開始位置がゼロの場合、 <i>instr</i> は文字列の最初の文字から検索を開始します。開始位置が正の数である場合、 <i>instr</i> は文字列の先頭からその数だけ数えた位置から検索を開始します。開始位置が負の数である場合、 <i>instr</i> は文字列の最後からその数だけ数えた位置から検索を開始します。この引数を省略すると、関数はデフォルト値の 1 を使用します。
<i>occurrence</i>	オプション	有効なトランスフォーメーション式を必要に応じて入力できます。検索値が文字列内に複数回出現する場合、何回目の出現を検索するかを指定できます。たとえば、開始位置から 2 回目に出現する検索値を検索する場合は、2 を入力します。 0 より大きい正の整数を入力できます。 この引数を省略すると、関数はデフォルト値の 1 を使用します。この場合、 <i>instr</i> は最初に出現する検索値を検索します。固定小数点値を渡すと、関数はそれを最も近い整数値に丸めます。負の整数または 0 を渡すと、関数は無効になります。
<i>comparison_type</i>	オプション	言語またはバイナリの文字列比較タイプ。 言語比較では言語固有の照合規則を考慮し、バイナリ比較ではビット単位の比較を行います。例えば、ドイツ語のシャープ文字である <i>s</i> は、言語比較では文字列 <i>"ss"</i> に一致しますが、バイナリ比較では一致しません。バイナリ比較は、言語比較よりも短い時間で実行されます。 次の整数値のいずれかを入力する必要があります。 - 0: <i>INSTR</i> は言語文字列比較を行います。 - 1: <i>INSTR</i> はバイナリ文字列比較を行います。 デフォルトは 0 です。

戻り値

検索が成功した場合は、整数。この整数は、*search* 因数の最初の文字が文字列内で左から数えて何文字目にあるかを示します。

検索に失敗した場合は、0 です。

instr に NULL 値を渡した場合は NULL になります。

例

次の式は、それぞれの会社名の先頭を基準に、文字‘a’が最初に出現する位置を返します。

```
sff:instr( COMPANY, 'a' )
```

次の表に、一部のサンプル値と戻り値を示します。

COMPANY	RETURN VALUE
Blue Fin Aqua Center	13
Maco Shark Shop	2
Scuba Gear	5
Frank's Dive Shop	3
VIP Diving Club	0

search 引数では大文字と小文字が区別されるため、‘Blue Fin Aqua Center’の‘A’をスキップして、‘Aqua’の‘a’の位置を返します。

次の式は、それぞれの会社名の先頭を基準に、文字‘a’が 2 回目に出現する位置を返します。

```
sff:instr( COMPANY, 'a', 1, 2 )
```

次の表に、一部のサンプル値と戻り値を示します。

COMPANY	RETURN VALUE
Blue Fin Aqua Center	0
Maco Shark Shop	8
Scuba Gear	9
Frank's Dive Shop	0
VIP Diving Club	0

search 引数では大文字と小文字が区別されるため、‘Blue Fin Aqua Center’の‘A’をスキップして、0 を返します。

次の式は、それぞれの会社名の最後の文字を基準に、文字‘a’が 2 回目に出現する位置を返します。

```
sff:instr( COMPANY, 'a', -1, 2 )
```

次の表に、一部のサンプル値と戻り値を示します。

COMPANY	RETURN VALUE
Blue Fin Aqua Center	0
Maco Shark Shop	2
Scuba Gear	5

COMPANY	RETURN VALUE
Frank's Dive Shop	0
VIP Diving Club	0

`search` 引数では大文字と小文字が区別されるため、'Blue Fin Aqua Center'の'A'をスキップして、0 を返します。

次の式は、それぞれの会社名の最後の文字を基準に、文字列'Blue Fin Aqua Center'で最初に出現する文字の位置を返します。

```
sff:instr( COMPANY, 'Blue Fin Aqua Center', -1, 1 )
```

次の表に、一部のサンプル値と戻り値を示します。

COMPANY	RETURN VALUE
Blue Fin Aqua Center	1
Maco Shark Shop	0
Scuba Gear	0
Frank's Dive Shop	0
VIP Diving Club	0

isNull

値が NULL であるかどうかを返します。

注: `isNull` 関数は、空の文字列を FALSE と評価します。

構文

```
util:isNull(value)
```

以下の表に、このコマンドの引数を示します。

引数	必須/ オプション	説明
値	必須	評価を行う行を渡します。有効なトランスフォーメーション式を必要に応じて入力できます。Binary 以外の任意のデータ型の値を渡すことができます。

戻り値

値が NULL の場合は、TRUE (1)。

値が NULL でない場合は、FALSE (0)。

例

次の例は、items テーブル内の NULL 値を確認します。

```
util:isNull( ITEM_NAME )
```

次の表に、一部のサンプル値と戻り値を示します。

ITEM_NAME	RETURN VALUE
Flashlight	0 (FALSE)
NULL	1 (TRUE)
Regulator system	0 (FALSE)
' '	0 (FALSE) <i>Empty string is not NULL</i>

lastDay

カラム内の各日付に対して、その月の最後の日の日付を返します。

構文

`date:lastDay(date)`

以下の表に、このコマンドの引数を示します。

引数	必須/ オプション	説明
<i>date</i>	必須	Date/Time データ型。月の最終日を返す日付を渡します。 日付を求める有効なトランスフォーメーション式を必要に応じて入力できます。

戻り値

日付。この関数に渡す日付値の月の最終日。

選択したカラム内の値が NULL である場合は、NULL。

例

次の式は、現在の日付を最終日として返します。

`date:lastDay(fn:current-dateTime('DATE'))`

次の表に、一部のサンプル値と戻り値を示します。

DATE	RETURN VALUE
18-04-98 01:00	Apr 18 1998 01:00:00 AM
20-08-99 05:00	Aug 20 1999 05:00:00 AM

次の式は、DATE カラム内の各日付に対して、月の最終日を返します。

`date:lastDay(date:addToDate(fn:current-dateTime('DATE','MM',-1))`

次の表に、一部のサンプル値と戻り値を示します。

DATE	RETURN VALUE
Apr 1 1998 12:00:00AM	Mar 31 1998 12:00:00AM

DATE	RETURN VALUE
Jan 6 1998 12:00:00AM	Dec 31 1997 12:00:00AM
Feb 2 1996 12:00:00AM	Jan 31 1996 12:00:00AM
NULL	NULL

toDate をネストして文字列値を日付に変換できます。toDate 関数には常に時間情報が含まれています。時刻値を含まない文字列を渡すと、返される日付には時刻「00:00:00」を含みます。

次の例は、各日付に対して、月の最終日を文字列と同じ形式で返します。

```
date:lastDay(toDate('DATE', 'MON-DD-YYYY'))
```

次の表に、一部のサンプル値と戻り値を示します。

DATE	RETURN VALUE
'18-NOV-98'	Nov-30-1998 00:00:00
'28-APR-98'	Apr-30-1998 00:00:00
NULL	NULL
'18-FEB-96'	Feb-29-1996 00:00:00 (<i>Leap year</i>)

```
date:lastDay(date:toDate("DATE", "YYYY-MM-DD"))
```

次の表に、一部のサンプル値と戻り値を示します。

DATE	RETURN VALUE
'18-NOV-98'	1998-Nov-30 00:00:00
'28-APR-98'	1998-Apr-30 00:00:00
NULL	NULL
'18-FEB-96'	1996-Feb-29 00:00:00 (<i>Leap year</i>)

lpad

文字列の先頭にいくつかの空白文字を追加して、文字列を指定した長さにします。

構文

```
sff:lpad(first_string, length, second_string)
```

次の表に、引数を示します。

引数	必須/ オプション	説明
<i>first_string</i>	必須	文字列。変更を行う文字列を渡します。有効なトランスフォーメーション式を必要に応じて入力できます。 NULL 値を渡すには、次の形式で空のシーケンスを指定する必要があります: ()
<i>length</i>	必須	正の整数リテラルでなければなりません。この引数によって、各文字列に必要な長さを指定します。 <i>length</i> が負の数である場合、 <i>lpad</i> は NULL を返します。
<i>second_string</i>	オプション	任意の文字列値。 <i>first_string</i> 値の左側に追加する文字列。有効なトランスフォーメーション式を必要に応じて入力できます。特定の文字列リテラルを入力できます。ただし、文字列の先頭に追加する文字は、'abc' のように一重引用符で囲みます。この引数は大文字と小文字を区別します。 <i>second_string</i> を省略すると、関数によって最初の文字列の先頭に空白文字が追加されます。 NULL 値を渡すには、次の形式で空のシーケンスを指定する必要があります: ()

戻り値

指定された長さの文字列。

関数に NULL 値を渡した場合、あるいは *length* が負の数の場合は、NULL です。

例

次の式は、数値の先頭にゼロを付加して、数値を 6 桁に標準化します。

```
sff:lpad(PART_NUM, 6, '0')
```

次の表に、一部のサンプル値と戻り値を示します。

PART_NUM	RETURN VALUE
702	000702
1	000001
0553	000553
484834	484834

lpad は、長さを左から数えます。最初の文字列が指定した長さよりも長い場合は、文字列が右側から切り詰められます。例えば、*lpad*('alphabetical', 5, 'x') は文字列 'alpha' を返します。

2 番目の文字列を加えた全文字数が、指定された長さに必要な文字数よりも長い場合は、2 番目の文字列の一部を使用した値が返されます。

```
sff:lpad(ITEM_NAME, 16, '*..*')
```

次の表に、一部のサンプル値と戻り値を示します。

ITEM_NAME	RETURN VALUE
Flashlight	*..**.Flashlight
Compass	*..**.*Compass
Regulator System	Regulator System

ITEM_NAME	RETURN VALUE
Safety Knife	*..*Safety Knife

次の式は、length 引数が負の数の場合に ITEM_NAME カラムの各行に対して lpad がどのような処理を行うかを示すものです。

```
sff:lpad(ITEM_NAME, -5, '.')
```

次の表に、一部のサンプル値と戻り値を示します。

ITEM_NAME	RETURN VALUE
Flashlight	NULL
Compass	NULL
Regulator System	NULL

ltrim

文字列の先頭から先頭のスペースまたは文字を削除します。

式で *trim_set* 引数を指定しない場合、ltrim は文字列の先頭からシングルスペースとダブルスペースを削除します。

ltrim を使用して文字列から文字を削除する場合、ltrim は *trim_set* を *str* 引数内の各文字と左から 1 文字ずつ比較します。文字列内の文字が *trim_set* 内のいずれかの文字と一致した場合、ltrim はその文字を削除します。ltrim 関数は、一致する文字が *trim_set* で見つからなくなるまで、文字を比較して削除します。その後、一致する文字が含まれない文字列を返します。

構文

```
sff:ltrim(str, trim_set)
```

次の表に、引数を示します。

引数	必須/ オプション	説明
<i>str</i>	必須	任意の文字列値。変更を行う文字列を渡します。任意の有効なトランスフォーメーション式を入力できます。文字列の先頭から文字を削除する前に、演算子を使って文字列の比較や連結を実行します。 文字列値は一重引用符または二重引用符で囲む必要があります。 NULL 値を渡すには、次の形式で空のシーケンスを指定する必要があります: ()
<i>trim_set</i>	オプション	任意の文字列値。文字列の先頭からの削除を行う文字を渡します。任意の有効なトランスフォーメーション式を入力できます。また、文字列を入力することもできます。 トリムセット値は一重引用符または二重引用符で囲む必要があります。 NULL 値を渡すには、次の形式で空のシーケンスを指定する必要があります: () ltrim 関数では大文字と小文字が区別されます。たとえば、文字列 'Alfredo' から文字 'A' を削除したい場合は、必ず 'a' ではなく 'A' と入力します。

戻り値

文字列。 *trim_set* 引数で指定された文字を削除した結果の文字列値。

`ltrim` に渡される値が `NULL` の場合は `NULL`。`trim_set` が `NULL` の場合、`ltrim` は `NULL` を返します。

例

次の式は、`LAST_NAME` カラムの文字列から文字‘`S`’と‘`.`’を削除します。

```
sff:ltrim( LAST_NAME, 'S.')
```

次の表に、一部のサンプル値と戻り値を示します。

LAST_NAME	RETURN VALUE
Nelson	Nelson
Osborne	Osborne
NULL	NULL
S. MacDonald	MacDonald
Sawyer	awyer
H. Bender	H. Bender
Steadman	teadman

`ltrim` 関数は、`S. MacDonald` から‘`S.`’を削除し、`Sawyer` および `Steadman` から‘`S`’を削除しますが、`H. Bender` からはピリオドを削除しません。これは、`ltrim` は `trim_set` 引数に指定された文字列を 1 文字ずつ検索していくためです。文字列内の最初の文字が `trim_set` 内の最初の文字と一致した場合、`ltrim` はその文字を削除します。その後、`ltrim` は文字列内の 2 番目の文字を確認します。その文字が `trim_set` 内の 2 番目の文字と一致している場合は削除し、3 文字目以降も同様に進みます。文字列内の最初の文字が `trim_set` の対応する文字と一致しない場合、`ltrim` はこの文字列を返して、次の行の評価を行います。

`H. Bender` の例では、`H` は `trim_set` 引数のどの文字にも一致しないため、`ltrim` は `LAST_NAME` カラム内の文字列を返して、次の行に移ります。

ltrim のヒント

2 つの文字列を連結した後に先頭の空白スペースを削除するには、`ltrim` を `CONCAT` とともに使用します。

`ltrim` をネストすることで、複数の文字セットを削除することもできます。例えば、名前のカラムから先頭の空白スペースと文字‘`T`’を削除するには、次のような式を作成します。

```
sff:ltrim( sff:ltrim( NAMES ), 'T' )
```

round (Numbers)

数値を指定の桁数または小数点以下の桁数に丸めます。また、`round` を使用して日付を丸めることもできます。

`round` 関数は次のように動作します。

- 引数に最も近い、端数のない数値を返します。
- 引数に近い数値が 2 つある場合は、正の無限大に最も近い数値が返されます。
- 引数のデータ型が `xs:float`、`xs:double`、`xs:decimal`、または `xs:integer` の 4 つの数値データ型のいずれかである場合、結果のデータ型は引数のデータ型と等しくなります。
- 引数のデータ型が数値データ型の 1 つから派生したデータ型である場合、結果は基本数値データ型のインスタンスになります。

構文

fn:round(arg)

以下の表に、このコマンドの引数を示します。

引数	必須/ オプション	説明
<i>arg</i>	必須	数値データ型で指定する必要があります。有効なトランスフォーメーション式を必要に応じて入力できます。値を丸める前に、演算子を使用して算術演算を実行できます。 NULL 値を渡すには、次の形式で空のシーケンスを指定する必要があります: () この関数は、数値の小数部分を切り捨てて、最も近い整数に丸めます。例えば、round(12.99, -)は 13 を返し、round(15.20, -)は 15 を返します。

戻り値

数値。

いずれかの引数が NULL の場合、round は NULL を返します。

例

次の式は、丸められた値を [Price] カラムに返します。

fn:round(PRICE)

次の表に、一部のサンプル値と戻り値を示します。

PRICE	RETURN VALUE
12.99	13.0
-15.99	-16.0
-18.99	-19.0
56.95	57.0
NULL	NULL

精度をより具体的に指定して、数値を最も近い整数に丸めたり、小数部分を切り捨てたりする場合は、round-half-to-even 関数を使用することをお勧めします。

fn:round-half-to-even(arg, precision)

正の精度を入力すると、関数は数値の小数点以下の桁数をこの値に丸めます。例えば、round(12.99, 1)は 13.0 を返し、round(15.44, 1)は 15.4 を返します。

負の精度を入力すると、関数は小数点の左側をこの桁数だけ丸めて、整数を返します。例えば、round(12.99, -1)は 10 を返し、round(15.99, -1)は 20 を返します。

返される値は、引数に最も近い値、つまり、負の精度の 10 の倍数である引数に数値的に最も近い値です。このような 2 つの値が非常に近い場合（例えば、引数の小数の部分がどちらも .500... である場合など）、関数は最下位桁が偶数である最も小さい値を返します。

例

次の式は、Price カラムの値を小数点以下 3 桁に丸めた値を返します。

```
fn:round-half-to-even(PRICE, 3)
```

次の表に、一部のサンプル値と戻り値を示します。

PRICE	RETURN VALUE
12.9936	12.994
15.9949	15.995
-18.8678	-18.868
56.9561	56.956
NULL	NULL

精度の引数に負の整数を渡すことにより、小数点の左側を指定桁数に丸めることもできます。

```
fn:round-half-to-even(PRICE, -2)
```

次の表に、一部のサンプル値と戻り値を示します。

PRICE	RETURN VALUE
13242.99	13200.0
1435.99	1400.0
-108.95	-100.0
NULL	NULL

精度の引数を省略すると、関数は数値を最も近い整数に丸めます。

```
fn:round-half-to-even(PRICE, 0)
```

次の表に、一部のサンプル値と戻り値を示します。

PRICE	RETURN VALUE
12.99	13.0
-15.99	-16.0
-18.99	-19.0
56.95	57.0
NULL	NULL

rtrim

文字列の末尾から空白文字または文字を削除します。

式で *trim_set* パラメータを指定しない場合、*rtrim* は文字列の末尾からシングルバイトスペースとダブルバイトスペースを削除します。

rtrim を使って文字列から文字を削除する場合、rtrim は *trim_set* を *string* 引数内の各文字と右から 1 文字ずつ比較します。文字列内の文字が *trim_set* 内のいずれかの文字と一致した場合、rtrim はその文字を削除します。rtrim は、一致する文字が *trim_set* で見つからなくなるまで、文字を比較して削除します。一致する文字を含まない文字列を返します。

構文

```
sff:rtrim(str, trim_set)
```

次の表に、引数を示します。

引数	必須/ オプション	説明
<i>string</i>	必須	任意の文字列値。切り詰めを行う値を渡します。任意の有効なトランスフォーメーション式を入力できます。文字列の末尾から空白文字を削除する前に、演算子を使って文字列の比較や連結を実行します。 文字列値は一重引用符または二重引用符で囲む必要があります。 NULL 値を渡すには、次の形式で空のシーケンスを指定する必要があります: ()
<i>trim_set</i>	オプション	任意の文字列値。文字列の末尾から削除を行う文字を渡します。文字列リテラルを入力することもできます。 文字列値は一重引用符または二重引用符で囲む必要があります。 NULL 値を渡すには、次の形式で空のシーケンスを指定する必要があります: () rtrim 関数では大文字と小文字が区別されます。例えば、文字列 'Alfredo' から 'o' という文字を削除する場合は、'O' ではなく 'o' と入力します。

戻り値

文字列。 *trim_set* 引数で指定された文字を削除した結果の文字列値。

rtrim に渡される値が NULL の場合は NULL。 *trim_set* が NULL の場合、rtrim は NULL を返します。

例

次の式は、LAST_NAME カラムの文字列から文字 're' を削除します。

```
sff:rtrim( LAST_NAME, 're')
```

次の表に、一部のサンプル値と戻り値を示します。

LAST_NAME	RETURN VALUE
Nelson	Nelson
Page	Pag
Osborne	Osborn
NULL	NULL
Sawyer	Sawy
H. Bender	H. Bend
Steadman	Steadman

`trim_set`の最初の文字は‘r’ですが、`rtrim`関数はPageから‘e’を削除します。これは、`rtrim`は`trim_set`引数に指定された文字列を1文字ずつ検索していくためです。文字列内の最後の文字が`trim_set`内の最初の文字と一致した場合、`rtrim`はその文字を削除します。ただし、文字列内の最後の文字が一致しない場合、`rtrim`は`trim_set`内の2番目の文字と比較します。文字列内の最後から2番目の文字が`trim_set`の2番目の文字と一致した場合、`rtrim`はその文字を削除します。文字列内の文字が`trim_set`と一致しなかった場合、`rtrim`はその文字列を返して、次の行の評価を行います。

最後の例では、Nelsonの最後の文字が`trim_set`引数のどの文字にも一致しないため、`rtrim`は‘Nelson’を返して次の行の評価します。

rtrim のヒント

2つの文字列を連結した後に末尾の空白文字を削除するには、`rtrim`を`CONCAT`とともに使用します。

また、`rtrim`をネストして複数の文字列を削除することもできます。例えば、名前のカラム内にあるそれぞれの文字列の末尾から末尾の空白と文字‘t’を削除する場合は、次のような式を作成します。

```
sff:trim( sff:rtrim( NAMES ), 't' )
```

toChar (Numbers)

数値をテキスト文字列に変換します。

構文

```
sff:toChar(xs:double(val))
```

注: 関数を追加した後に、構文にフレーズ（`xs:double`）を手動で追加する必要があります。フレーズを追加しない場合、タスクフローは失敗します。

次の表に、引数を示します。

引数	必須/オプション	説明
<i>val</i>	必須	文字列への変換を行う数値。有効なトランスフォーメーション式を必要に応じて入力できます。

`toChar`は、次のように倍精度浮動小数点数をテキスト文字列に変換します。

- 16桁までの倍精度浮動小数点数値を文字列に変換し、15桁までの精度を提供します。15桁を超える数値を渡すと、`toChar`は数値を16桁に基づいて丸め、数値の文字列表現を科学的表記で返します。例えば、1234567890123456倍精度浮動小数点数値は、‘1.23456789012346e+015’文字列値に変換されます。
- (-1e16,-1e-16]および[1e-16,1e16)の範囲の数値は、10進表記で返します。`toChar`は、この範囲以外の数値を科学的表記で返します。例えば、10842764968208837340倍精度浮動小数点数値は、‘1.08427649682088e+019’文字列値に変換されます。

`toChar`は、次のように10進数値をテキスト文字列に変換します。

- 低精度モードの場合、`toChar`は、固定小数点値を倍精度浮動小数点数値として扱います。
- 小数点以下を`toChar`関数に渡し、入力値に小数点以下のスケールと一致させるための桁が足りない場合、`toChar`関数は値にゼロを追加します。

例えば、小数点以下のスケールが5で、行の値が7.6901の場合、`toChar`関数は入力値を7.69010として扱い、戻り値は「7.69010」になります。

戻り値

文字列。

関数に NULL 値を渡した場合は NULL です。

倍精度浮動小数点数の変換例

次の式は、SALES ポート内の倍精度浮動小数点数値を文字列に変換します。

```
sff:toChar(xs:double(SALES))
```

SALES	RETURN VALUE
1010.99	'1010.99'
-15.62567	'-15.62567'
10842764968208837340	'1.08427649682088e+019' (rounded based on the 16th digit and returns the value in scientific notation)
236789034569723	'236789034569723'
0	'0'
33.15	'33.15'
NULL	NULL

10 進数値の変換例

次の式は、SALES ポート内の 10 進数値を、高精度モードで文字列に変換します。

```
sff:toChar(xs:double(SALES))
```

次の表に、精度が 38 より大きい場合のサンプル値と戻り値を示します。

SALES	RETURN VALUE
2378964536789761	'2378964536789761'
1234567890123456789012345679	'1234567890123456789012345679'
1.234578945469649345876123456	'1.234578945469649345876123456'
0.9999999999999999999999999999	'0.9999999999999999999999999999'
12345678901234567890123456799	'12345678901234567890123456799'
23456788992233456678458934567123465239	'23456788992233456678458934567123465239'
423456789012345678901234567991234567899	'4.23456789012346e+038'
(38 より大きい)	

次の表に、精度が 28 より大きい場合のサンプル値と戻り値を示します。

SALES	RETURN VALUE
2378964536789761	'2378964536789761'
1234567890123456789012345679	'1234567890123456789012345679'
1.234578945469649345876123456	'1.234578945469649345876123456'

次の表に、一部のサンプル値と戻り値を示します。

DATE_PROMISED	RETURN VALUE
'01/22/98'	Jan 22 1998 00:00:00
'05/03/98'	May 3 1998 00:00:00
'11/10/98'	Nov 10 1998 00:00:00
'10/19/98'	Oct 19 1998 00:00:00
NULL	NULL

`date:toDate(xs:dateTime('DATE_PROMISED'), 'MON DD YYYY HH12:MI:SSAM')`

次の表に、一部のサンプル値と戻り値を示します。

DATE_PROMISED	RETURN VALUE
'Jan 22 1998 02:14:56PM'	Jan 22 1998 02:14:56PM
'Mar 15 1998 11:11:11AM'	Mar 15 1998 11:11:11AM
'Jun 18 1998 10:10:10PM'	Jun 18 1998 10:10:10PM
'October 19 1998'	<i>Error. Integration Service skips this row.</i>
NULL	NULL

以下の式は、SHIP_DATE_MJD_STRING ポートの文字列を日付値に変換します。

`date:toDate(xs:dateTime('SHIP_DATE_MJD_STR'), 'J')`

次の表に、一部のサンプル値と戻り値を示します。

SHIP_DATE_MJD_STR	RETURN VALUE
'2451544'	Dec 31 1999 00:00:00.000000000
'2415021'	Jan 1 1900 00:00:00.000000000

J フォーマット文字列には日付の時間部分が含まれないため、戻り値では時間が 00.000000000:00:00 に設定されています。

次の式は、文字列を 4 桁形式の年に変換します。現在の年は 1998 です。

`date:toDate(xs:dateTime('DATE_STR'), 'MM/DD/RR')`

次の表に、一部のサンプル値と戻り値を示します。

DATE_STR	RETURN VALUE
'04/01/98'	04/01/1998 00:00:00.000000000
'08/17/05'	08/17/2005 00:00:00.000000000

次の式は、文字列を 4 桁形式の年に変換します。現在の年は 1998 です。

```
date:toDate(xs:dateTime('DATE_STR'), 'MM/DD/YY')
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_STR	RETURN VALUE
'04/01/98'	04/01/1998 00:00:00.000000000
'08/17/05'	08/17/1905 00:00:00.000000000

注: 2 番目の行については、RR は 2005 年を返しますが、YY は 1905 年を返します。

次の式は、文字列を 4 桁形式の年に変換します。現在の年は 1998 です。

```
date:toDate(xs:dateTime('DATE_STR'), 'MM/DD/Y')
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_STR	RETURN VALUE
'04/01/8'	04/01/1998 00:00:00.000000000
'08/17/5'	08/17/1995 00:00:00.000000000

次の式は、文字列を 4 桁形式の年に変換します。現在の年は 1998 です。

```
date:toDate(xs:dateTime('DATE_STR'), 'MM/DD/YYYY')
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_STR	RETURN VALUE
'04/01/998'	04/01/1998 00:00:00.000000000
'08/17/995'	08/17/1995 00:00:00.000000000

次の式は、深夜からの通算秒を含む文字を日付値に変換します。

```
date:toDate(xs:dateTime('DATE_STR'), 'MM/DD/YYYY SSSS')
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_STR	RETURN VALUE
'12/31/1999 3783'	12/31/1999 01:02:03
'09/15/1996 86399'	09/15/1996 23:59:59

toDecimal

文字列または数値を 10 進値に変換します。toDecimal 関数は、先頭のスペースを無視します。

構文

```
util:toDecimal(value, scale)
```


次の表に、引数を示します。

引数	必須/ オプション	説明
値	必須	String または Numeric データ型で指定する必要があります。10 進数に変換する値を渡します。 有効なトランスフォーメーション式を必要に応じて入力できます。
スケール	オプション	0 から 28 までの整数リテラル。小数点以下の桁数を指定します。この引数を省略すると、関数は入力値と同じ位取りの値を返します。 この引数は、詳細モードで必要です。

戻り値

0 から 28 までの精度と位取りを持つ 10 進値。

選択したカラムの値が空の文字列または数値以外の文字であった場合は、ゼロ。

関数に NULL 値を渡した場合は NULL です。

例

次の式は、カラム IN_TAX からの値を使用します。データ型は Decimal で、精度は 10、スケールは 3 です。

```
util:toDecimal(IN_TAX, 3)
```

次の表に、一部のサンプル値と戻り値を示します。

IN_TAX	RETURN VALUE
'15.6789'	15.679
'60.2'	60.200
'118.348'	118.348
NULL	NULL
'A12.3Grove'	0

次の式は、[Sales] カラムからの値を使用します。データ型は Decimal で、精度は 10、スケールは 2 です。

```
util:toDecimal(Sales, 2)
```

次の表に、一部のサンプル値と戻り値を示します。

Sales	RETURN VALUE
'1234'	1234
'1234.01'	1234.01

1234.00 形式の戻り値が必要な場合は、次の式を使用します。

```
format-number(util:toDecimal('1234', 2), '0.00')
```

toDecimal 関数は、最大 28 桁の精度をサポートします。32 桁など、28 桁を超える精度で値を渡した場合は、高精度を有効にした後でも問題が発生します。

ソースからターゲットまで 32 桁の値をそのまま使用する場合は、32 桁の精度でソースからターゲットまでの文字列として値を定義する必要があります。ターゲットデータベースに数値データ型があり、ターゲット定義に文字列データ型がある場合でも、ターゲットデータベースに値をロードできます。ただし、ターゲットデータベースのカラムのデータ型が 32 桁の精度を受け入れることを確認する必要があります。

10 進数のオーバーフロー

小数点の左側の数値のサイズが精度を超えた場合は、10 進数の演算がオーバーフローします。

この問題を解決するには、式ポートのスケールや精度を変更し、マッピングのダウストリームポートを接続して、式の入力データのサイズに合わせた設定を使用します。

以下に例を示します。

数値フィールドのスケールが 15 に定義され、サイズが 28 に定義されている場合、このフィールドは小数点の左側に 13 個、右側に 15 個の数値を受け入れます。したがって、次の数値は 28,15 という数値に対して有効になります。

1234567890123.11143

13575.123451234567891

ただし、次の数値は 10 進数のオーバーフローエラーを引き起こします。

11111222233333.4444

123.111112222333334

toInteger

文字列または数値を整数に変換します。toInteger 構文にはオプションの引数があり、数字を近似値の整数に丸めるか、小数点以下を切り詰めるかを選択できます。toInteger 関数は、先頭のスペースを無視します。

構文

util:toInteger(*value*, *flag*)

次の表に、引数を示します。

引数	必須/ オプション	説明
<i>value</i>	必須	String または Numeric データ型で指定する必要があります。整数に変換する値を渡します。 有効なトランスフォーメーション式を必要に応じて入力できます。
<i>flag</i>	オプション	小数点以下を切り捨てるか丸めるかを指定します。フラグは整数リテラルか、TRUE または FALSE の定数でなければなりません。 - toInteger は、フラグが TRUE または 0 以外の数字の場合に小数点以下を切り捨てます。 - フラグが FALSE または 0 の場合、あるいはこの引数が省略された場合、toInteger は値を近似値の整数に丸めます。

戻り値

整数。

関数に NULL 値を渡した場合は NULL です。

関数に渡された値に英数字が含まれている場合は 0 です。

例

次の式は、カラム IN_TAX からの値を使用します。

```
util:toInteger(IN_TAX, fn:boolean(1))
```

次の表に、一部のサンプル値と戻り値を示します。

IN_TAX	RETURN VALUE
'15.6789'	15
'60.2'	60
'118.348'	118
NULL	NULL
'A12.3Grove'	0
' 123.87'	123
'-15.6789'	-15
'-15.23'	-15

bigint カラムが整数データ型のカラムにマッピングされている場合は問題が発生します。

この問題を回避するには、マッピング全体でカラムのデータ型を等しくする必要があります。整数カラムに bigint を割り当てる必要がある場合は、渡されるデータが整数の範囲を超えないようにしてください。

bigint の範囲: -9,223,372,036,854,775,808～9,223,372,036,854,775,807

整数の範囲: -2,147,483,648～2,147,483,647

trunc

日付を特定の年、月、日、時、分、秒、またはミリ秒に切り詰めます。また、trunc を使って数値を切り詰めることもできます。

日付の中の以下の部分を切り詰めることができます。

- **年。**日付の年の部分を切り詰めると、関数は入力された年の 1 月 1 日を返し、時刻を 00:00:00.000000000 に設定します。例えば、次の式では 1/1/1997 00:00:00.000000000 を返します。
`date:trunc(xs:dateTime('12/1/1997 3:10:15'), 'YY')`
- **月。**日付の月の部分を切り詰めると、関数は月の最初の日を返し、時刻を 00:00:00.000000000 に設定します。例えば、次の式では 4/1/1997 00:00:00.000000000 を返します。
`date:trunc(xs:dateTime('4/15/1997 12:15:00'), 'MM')`
- **日。**日付の日の部分を切り詰めると、関数はその日付を返し、時刻を 00:00:00.000000000 に設定します。例えば、次の式では 6/13/1997 00:00:00.000000000 を返します。
`date:trunc(xs:dateTime('6/13/1997 2:30:45'), 'DD')`
- **時間。**日付の時の部分を切り詰めると、この関数は日付の分と秒をゼロに設定して返します。たとえば、次の式では 4/1/1997 11:00:00.000000000 を返します。
`date:trunc(xs:dateTime('4/1/1997 11:29:35'), 'HH')`
- **分。**日付の分の部分を切り詰めると、この関数は日付の秒をゼロに設定して返します。たとえば、次の式では 5/22/1997 10:15:00.000000000 を返します。
`date:trunc(xs:dateTime('5/22/1997 10:15:29'), 'MI')`

- **秒**。日付の分の部分を切り詰めると、この関数は日付の秒をゼロに設定して返します。たとえば、次の式では 5/22/1997 10:15:29.000000000 を返します。

```
date:trunc(xs:dateTime('5/22/1997 10:15:29.135'), 'SS')
```

- **ミリ秒**。日付の分の部分を切り詰めると、この関数は日付の秒をゼロに設定して返します。たとえば、次の式では 5/22/1997 10:15:30.135000000 を返します。

```
date:trunc(xs:dateTime('5/22/1997 10:15:30.135235'), 'MS')
```

構文

```
date:trunc(xs:dateTime('date'), 'format')
```

注: xs:dateTime 句を手動で追加し、日付値を一重引用符で囲む必要があります。

次の表に、引数を示します。

引数	必須/ オプション	説明
<i>日付</i>	必須	Date/Time データ型。切り詰めを行う日付値です。日付を求める任意の有効なトランスフォーメーション式を入力できます。 NULL 値を渡すには、次の形式で空のシーケンスを指定する必要があります: ()
<i>format</i>	必須	正しいフォーマット文字列を入力します。フォーマット文字列は大文字と小文字を区別しません。 NULL 値を渡すには、次の形式で空のシーケンスを指定する必要があります: ()

戻り値

日付。

関数に NULL 値を渡した場合は NULL です。

例

以下の式は、DATE_SHIPPED カラムの日付の年の部分を切り詰めます。

```
date:trunc(xs:dateTime('DATE_SHIPPED'), 'Y')
date:trunc(xs:dateTime('DATE_SHIPPED'), 'YY')
date:trunc(xs:dateTime('DATE_SHIPPED'), 'YYY')
date:trunc(xs:dateTime('DATE_SHIPPED'), 'YYYY')
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 1 1998 00:00:00.000000000
Apr 19 1998 1:31:20PM	Jan 1 1998 00:00:00.000000000
Jun 20 1998 3:50:04AM	Jan 1 1998 00:00:00.000000000
Dec 20 1998 3:29:55PM	Jan 1 1998 00:00:00.000000000
NULL	NULL

以下の式は、DATE_SHIPPED カラムの各日付の月の部分を切り詰めます。

```
date:trunc(xs:dateTime('DATE_SHIPPED'), 'MM')
date:trunc(xs:dateTime('DATE_SHIPPED'), 'MON')
date:trunc(xs:dateTime('DATE_SHIPPED'), 'MONTH')
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 1 1998 00:00:00.000000000
Apr 19 1998 1:31:20PM	Apr 1 1998 00:00:00.000000000
Jun 20 1998 3:50:04AM	Jun 1 1998 00:00:00.000000000
Dec 20 1998 3:29:55PM	Dec 1 1998 00:00:00.000000000
NULL	NULL

以下の式は、DATE_SHIPPED カラムの各日付の日の部分を切り詰めます。

```
date:trunc(xs:dateTime('DATE_SHIPPED'), 'D')
date:trunc(xs:dateTime('DATE_SHIPPED'), 'DD')
date:trunc(xs:dateTime('DATE_SHIPPED'), 'DDD')
date:trunc(xs:dateTime('DATE_SHIPPED'), 'DV')
date:trunc(xs:dateTime('DATE_SHIPPED'), 'DAY')
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 15 1998 00:00:00.000000000
Apr 19 1998 1:31:20PM	Apr 19 1998 00:00:00.000000000
Jun 20 1998 3:50:04AM	Jun 20 1998 00:00:00.000000000
Dec 20 1998 3:29:55PM	Dec 20 1998 00:00:00.000000000
Dec 31 1998 11:59:59PM	Dec 31 1998 00:00:00.000000000
NULL	NULL

以下の式は、DATE_SHIPPED カラムの各日付の時の部分を切り詰めます。

```
date:trunc(xs:dateTime('DATE_SHIPPED'), 'HH')
date:trunc(xs:dateTime('DATE_SHIPPED'), 'HH12')
date:trunc(xs:dateTime('DATE_SHIPPED'), 'HH24')
```

次の表に、一部のサンプル値と戻り値を示します。

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:31AM	Jan 15 1998 02:00:00.000000000
Apr 19 1998 1:31:20PM	Apr 19 1998 13:00:00.000000000
Jun 20 1998 3:50:04AM	Jun 20 1998 03:00:00.000000000

DATE_SHIPPED	RETURN VALUE
Dec 20 1998 3:29:55PM	Dec 20 1998 15:00:00.000000000
Dec 31 1998 11:59:59PM	Dec 31 1998 23:00:00.000000000
NULL	NULL

以下の式は、DATE_SHIPPED カラムの各日付の分の部分を切り詰めます。

```
date:trunc(xs:dateTime('DATE_SHIPPED'), 'MI')
```



次の表に、一部のサンプル値と戻り値を示します。

DATE_SHIPPED	RETURN VALUE
Jan 15 1998 2:10:30AM	Jan 15 1998 02:10:00.000000000
Apr 19 1998 1:31:20PM	Apr 19 1998 13:31:00.000000000
Jun 20 1998 3:50:04AM	Jun 20 1998 03:50:00.000000000
Dec 20 1998 3:29:55PM	Dec 20 1998 15:29:00.000000000
Dec 31 1998 11:59:59PM	Dec 31 1998 23:59:00.000000000
NULL	NULL

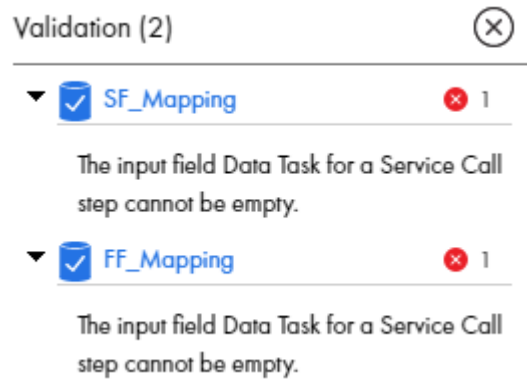
[検証] パネルの使用

[検証] パネルには、タスクフローのエラーが一覧表示されます。[検証] パネルを使用して、タスクフローのトラブルシューティングを行います。

1. [Taskflow Designer] ページの右上で、**[検証]** をクリックします。
タスクフローにエラーがある場合は、エラーの一覧が表示されます。

Validation (2) ⓧ	
▶  SF_Mapping	✖ 1
▶  FF_Mapping	✖ 1

2. 各エラーを展開して詳細を表示します。



3. エラーをクリックすると、UI の対応する手順に進みます。
4. タスクフローに変更を加え、**【保存】** をクリックします。

タスクフローの実行

タスクフローを使用して、複数のデータ統合タスクの実行シーケンスを制御します。

次の方法で、タスクフローを呼び出して実行できます。

タスクフローデザイナーから

タスクフローをタスクフローデザイナーから実行するには、タスクフローを開き、ページの右上で **【実行】** をクリックします。

また、1 つ以上のタスクフロー入力を作成し、その入力を使用してタスクフローを実行することもできます。タスクフロー入力を使用してタスクフローを実行するには、タスクフローを開き、**【アクション】** メニューで **【次を使用して実行】** を選択します。

API として

タスクフローを API として実行するには、最初にタスクフローをサービスとしてパブリッシュしてから実行する必要があります。タスクフローをパブリッシュする際、データ統合はサービス URL と SOAP サービス URL を生成します。これらのエンドポイント URL を使用して、タスクフローを API として実行できます。タスクフローを API として実行する際は、タスクフローに含まれるタスクの入力パラメータを動的に指定して、オーケストレーションを実行できます。

RunAJob ユーティリティの使用

RunAJob ユーティリティを使用してタスクフローを実行するには、タスクフローをパブリッシュする必要があります。RunAJob ユーティリティを使用するには、RunAJob ユーティリティコマンド `cli.bat runAJobCli` の後に引数を入力します。

ファイルリスナによる開始

コネクタファイルリスナ経由でタスクフローを呼び出す事ができます。タスクフロー内で、バインディングタイプを **【イベント】** として定義し、イベントソースとしてコネクタファイルリスナを選択します。タスクフローをパブリッシュするときに、タスクフローは其中で定義されているコネクタファイルリスナにサブスクライブします。ファイルイベントが発生すると、コネクタファイルリスナはタスクフローを呼び出します。例えば、フォルダ上の新しいファイルをリスンするようにコネクタファイルリスナを設定した場合、コネクタファイルリスナは指定したフォルダに新しいファイルが到着するたびに、関連するタスクフローを呼び出します。

ファイルリサナの詳細については、『コンポーネント』を参照してください。

スケジュールに従う

スケジュールに従ってタスクフローを実行するには、Administrator でスケジュールを作成し、タスクフローをそのスケジュールに関連付けます。

スケジュールの詳細については、Administrator のヘルプの「スケジュール」を参照してください。

タスクフローデザイナーからのタスクフローの実行

タスクフローデザイナーからタスクフローを実行するには、タスクフローの読み取りおよび実行権限が必要です。

1. **データ統合**で、左側のナビゲーションペインから **【参照】** をクリックします。
2. **【参照基準】** 一覧で、**【アセット】** を選択します。
3. **【すべてのアセット】** > **【タスクフロー】** に移動します。
4. 実行するタスクフローをクリックします。
5. ページの右上で **【実行】** をクリックします。

タスクフローの実行タスクフロー入力の使用

タスクフローをパブリッシュした後に、1つ以上のタスクフロー入力を作成し、そのタスクフロー入力を使用してタスクフローを実行することで、タスクフロー入力をテストすることができます。タスクフローを実行した後に、タスクフローインスタンスの実行の成功と失敗に関する詳細を表示できます。

タスクフロー入力を使用してタスクフローを実行するには、タスクフローデザイナーの **【許可されたユーザー】** および **【許可されたグループ】** に値を含める必要があります。

タスクフローの作成入力

タスクフローをパブリッシュした後に、タスクフロー入力を作成し、その入力を使用してタスクフローをテスト目的で実行できます。複数のタスクフロー入力を作成し、すべての入力を使用してタスクフローを実行することもできます。

タスクフロー入力は、JSON 形式または XML 形式で作成できます。次に、タスクフロー入力を検証して保存します。

タスクフロー入力を含むタスクフローをエクスポート、コピー、または移動すると、タスクフロー入力が保持されます。

1. **【参照】** ページで、タスクフロー入力を作成するタスクフローに移動します。
2. **【アクション】** メニューで **【次を使用して実行】** を選択します。

【タスクフロー入力コレクションのテスト】 ページが開きます。

以下の図に、**【タスクフロー入力コレクションのテスト】** ページを示します。



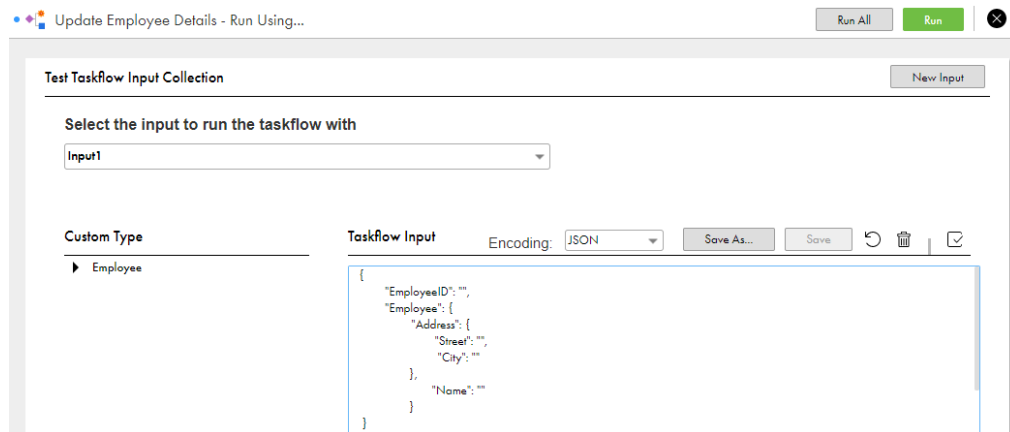
3. **【新しい入力】** をクリックします。
4. タスクフロー入力の名前を入力して、**【保存】** をクリックします。

80 文字を超える名前は使用できません。

【タスクフロー入力】 セクションには、タスクフローに必要な入力フィールドが読み込まれます。

注: カスタムタイプの入力フィールドの場合は、デフォルトで、入力ペイロードにフォールトと出力パラメータが追加され、**【タスクフロー入力】** セクションに表示されます。タスクフロー入力を有効にするには、それらのパラメータを手動で削除する必要があります。

以下の図は、**【従業員】** カスタムタイプの入力フィールドが読み込まれた **【タスクフロー入力】** セクションを示しています。



【カスタムタイプ】 セクションには、タスクフローが使用するカスタムタイプの入力フィールドが表示されます。カスタムタイプフィールドの詳細については、[Creating an input field with a custom type \(ページ 18\)](#) を参照してください。

5. **【エンコーディング】** リストから、使用する形式に基づいて **【JSON】** または **【XML】** を選択し、タスクフロー入力を指定します。

注: タスクフローにスペース文字を含む入力フィールドが含まれている場合は、エンコーディングタイプに **【JSON】** を選択します。

6. タスクフロー入力の構文を検証するには、**【検証】** アイコンをクリックします。

検証が成功したかどうかを示す確認メッセージが表示されます。検証が失敗した場合はタスクフロー入力の構文を修正し、再度検証します。

7. タスクフロー入力を保存するには、**【保存】** をクリックします。

タスクフロー入力に別の名前を付けて保存するには、**【名前を付けて保存】** をクリックします。また、タスクフロー入力を最後に保存した状態にリセットする場合は、**【リセット】** アイコンをクリックします。

プロセス入力を作成した後に、必要な入力を使用してプロセスを実行できます。

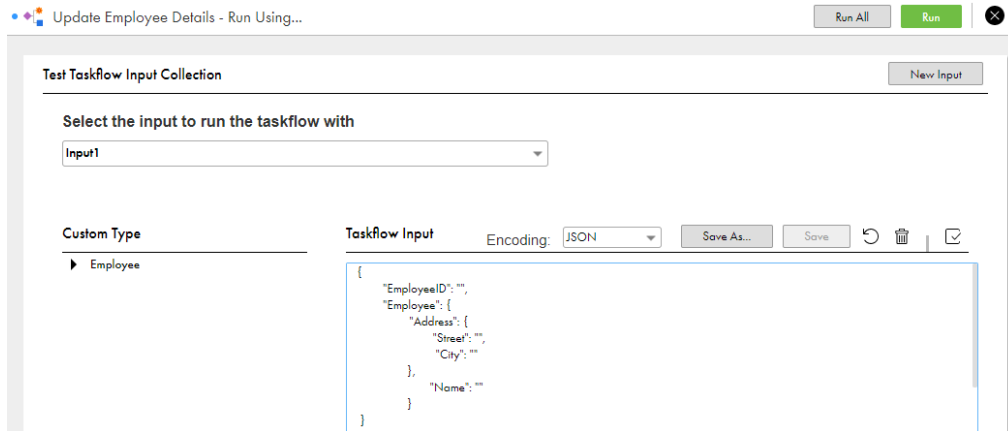
タスクフロー入力によるタスクフローの実行

タスクフローをパブリッシュしてタスクフロー入力を作成した後に、そのタスクフロー入力を使用してタスクフローを実行することで、タスクフロー入力をテストすることができます。タスクフローを実行した後に、タスクフローの実行の詳細を確認できます。

1. **【参照】** ページで、1 つ以上のタスクフロー入力を使用して実行するタスクフローに移動します。
2. 特定のタスクフロー入力またはすべてのタスクフロー入力を使用してタスクフローを実行するには、**【アクション】** メニューで **【次を使用して実行】** を選択します。

【タスクフロー入力コレクションのテスト】 ページが開きます。

以下の図に、**【タスクフロー入力コレクションのテスト】** ページを示します。

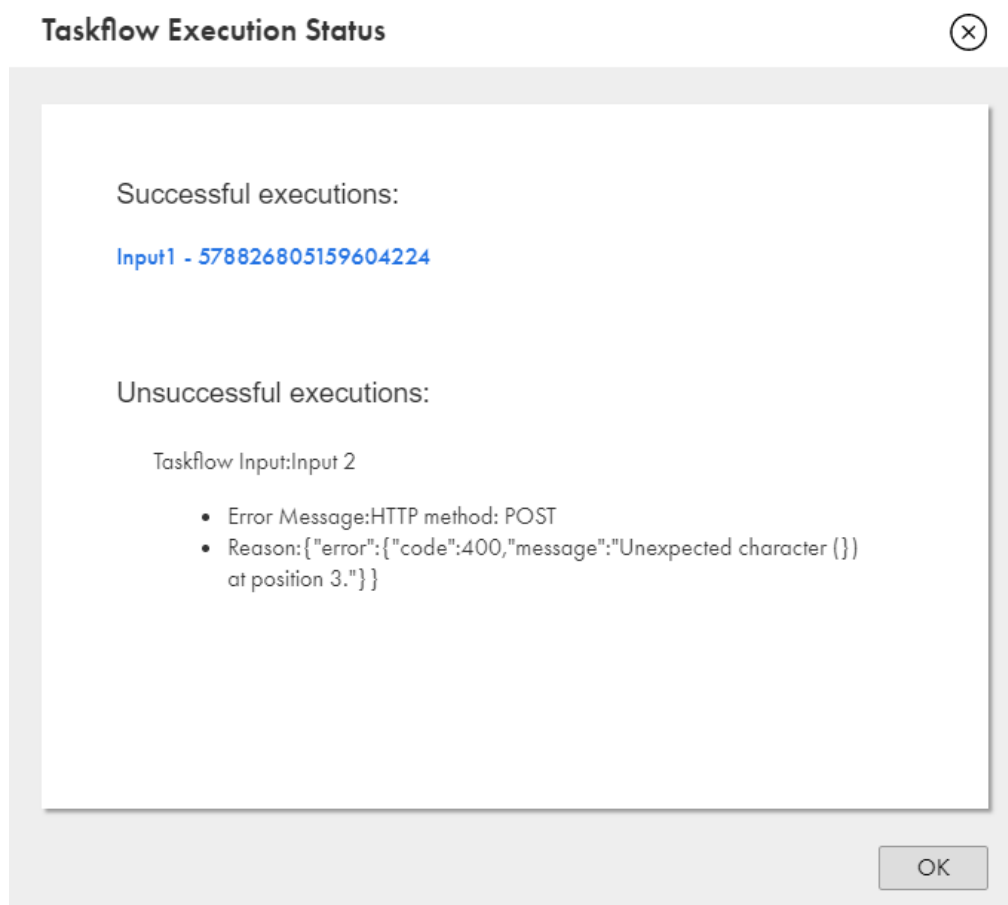


注: 保存されていない変更やパブリッシュされていない変更がタスクフローに含まれている場合、**【次を使用して実行】** オプションは無効になります。タスクフローを実行するには、タスクフローを保存してパブリッシュする必要があります。

3. 次のいずれかの手順に従います。
 - 特定のタスクフロー入力を使用してタスクフローを実行するには、タスクフロー入力を選択し、**【実行】** をクリックします。
 - すべてのタスクフロー入力を使用してタスクフローを実行するには、**【すべて実行】** をクリックします。

データ統合は、指定された入力でタスクフローを実行します。タスクフロー実行が完了すると **【タスクフロー実行ステータス】** ページが開き、タスクフローの実行の成功と失敗に関する詳細が表示されます。長時間実行されるタスクフローの場合は、**【タスクフロー実行ステータス】** ページが表示されるまでに時間がかかることがあります。

以下の図は、**【タスクフロー実行ステータス】** ページを示しています。



4. タスクフロー実行の詳細を表示するには、**【実行成功】** セクションまたは **【実行失敗】** セクションのリンクをクリックします。

タスクフロー実行 ID がリンクに追加されます。実行 ID を使用して、**【マイジョブ】** ページでタスクフローの実行の詳細を確認できます。

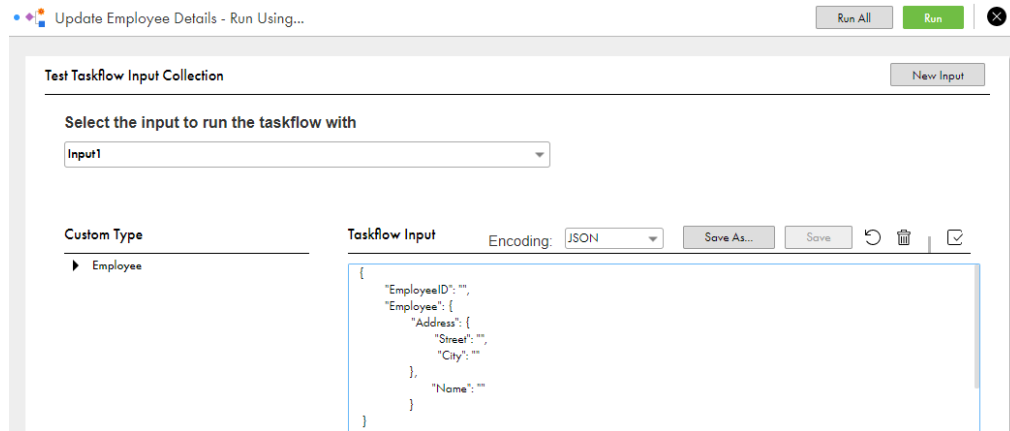
タスクフロー入力の削除

不要なタスクフロー入力は削除できます。

1. **【参照】** ページで、タスクフロー入力を削除するタスクフローに移動します。
2. **【アクション】** メニューで **【次を使用して実行】** を選択します。

【タスクフロー入力コレクションのテスト】 ページが開きます。

以下の図に、**【タスクフロー入力コレクションのテスト】** ページを示します。



3. **【タスクフロー実行に使用する入力を選択】** フィールドから削除するタスクフロー入力を選択し、**【削除】** アイコンをクリックします。
タスクフロー入力の削除の確認を求めるメッセージが表示されます。
4. 削除を続行するには **【削除】**、削除をキャンセルするには **【キャンセル】** をクリックします。

タスクフローのパブリッシュ

タスクフローを API として実行したり、タスクフローをスケジュールしたりする前に、タスクフローをサービスとしてパブリッシュする必要があります。

タスクフローをパブリッシュする際、データ統合はサービス URL と SOAP サービス URL を生成します。これらのエンドポイント URL を使用して、タスクフローを API として実行できます。タスクフローを API として実行する際は、タスクフローに含まれるタスクの入力パラメータを動的に指定して、オーケストレーションを実行できます。

注: タスクフローデザイナからタスクフローを実行すると、データ統合では自動的にタスクフローがパブリッシュされ、サービス URL と SOAP サービス URL が生成されます。

パブリッシュ済みのタスクフローを編集した場合は、タスクフローを再度パブリッシュして、変更内容がタスクフロー API に反映されるようにする必要があります。

サービスとしてパブリッシュされたタスクフロー API を無効にするには、タスクフローをパブリッシュ解除する必要があります。必要な変更を行った後で、タスクフローを再度パブリッシュして、変更内容がタスクフロー API に反映されるようにします。

1. データ統合で **【参照】** を選択します。
【参照】 ページが開きます。
2. サービスとしてパブリッシュするタスクフローに移動します。
3. ページの右上にある **【アクション】** メニューから **【パブリッシュ】** をクリックします。

データ統合は、タスクフローをサービスとしてパブリッシュします。

注: タスクフローをパブリッシュ解除するには、**【アクション】** メニューで **【パブリッシュ解除】** を選択します。

複数のタスクフローをまとめてパブリッシュすることもできます。詳細については、[「タスクフローの一括パブリッシュ」 \(ページ 109\)](#)を参照してください。

タスクフローの一括パブリッシュ

publish リソースを使用すると、複数のタスクフローを同時にパブリッシュして時間を節約できます。

1. REST クライアントでは、次の URI の POST 要求を使用します。
<Informatica Intelligent Cloud Services の URL>/active-bpel/asset/v1/publish
2. 次のヘッダーを追加します。

キー	値
承認	application/vnd.api+json
Content-Type	application/vnd.api+json
INFA-SESSION-ID	login リソースを使用して、セッション ID を取得します。ログインリソースの詳細は、『 <i>REST API リファレンス</i> 』を参照してください。

3. 本文で assetPaths 属性を使用して、パブリッシュするタスクフローの場所と名前を指定します。
次の形式を使用します。

```
{
  "data": {
    "type": "publish",
    "attributes": {
      "assetPaths": [
        "Explore/<location-of-taskflow1>/<name-of-taskflow1>.TASKFLOW.xml",
        "Explore/<location of taskflow2>/<name-of-taskflow2>.TASKFLOW.xml",
        "Explore/<location-of-taskflown>/<name-of-taskflown>.TASKFLOW.xml"
      ]
    }
  }
}
```

4. POST 要求を送信します。

パブリッシュ ID と、成功応答または失敗応答が表示されます。要求が失敗した場合、応答にはエラーの詳細も含まれます。

次のスニペットは、応答の例を示しています。

```
{
  "data": {
    "type": "publish",
    "id": "690487059198201856",
    "attributes": {
      "jobState": "NOT_STARTED",
      "jobStatusDetail": {},
      "startedBy": "autouser_pod1",
      "startDate": "2022-03-21T09:09:04.000+0000",
      "totalCount": 1,
      "processedCount": 0,
      "assetPaths": [
        "Explore/Pavan/BulkPublishApi/BPTaskflow1.TASKFLOW.xml"
      ]
    }
  },
  "links": {
    "self": "https://na1.dm-us.informaticacloud.com/active-bpel/asset/v1/publish/690487059198201856",
    "status": "https://na1.dm-us.informaticacloud.com/active-bpel/asset/v1/publish/690487059198201856/Status"
  }
}
```

この例のパブリッシュ ID は 690487059198201856 です。

5. パブリッシュのステータスとパブリッシュジョブに関する情報を表示するには、次の URL の GET 要求を使用します。

URL	説明
<Informatica Intelligent Cloud Services の URL>/active-bpel/asset/v1/publish/<publishId>/Status	パブリッシュのステータスを表示します。
<Informatica Intelligent Cloud Services の URL>/active-bpel/asset/v1/publish/<publishId>	パブリッシュジョブの情報を表示します。

API としてのタスクフローの実行

タスクフローをサービスとしてパブリッシュした後、タスクフローを API として実行したり、タスクフローをスケジュールしたりできます。エンドポイント URL を使用して、タスクフローを API として実行し、タスクフローに含まれるタスクの入力パラメータを動的に指定して、オーケストレーションを実行できます。

1. サービスとしてパブリッシュしたタスクフローに移動します。
2. **【アクション】** メニューで **【プロパティの詳細】** を選択します。

次の図に示すように、**【プロパティの詳細】** ダイアログボックスにサービス URL と SOAP サービス URL が表示されます。

The image shows a dialog box titled "Properties Detail for TaskflowMultiTasks". It has a close button (X) in the top right corner. The dialog is divided into two main sections: "Basic" and "Endpoints".

Basic Section:

- Unique Name: TaskflowMultiTasks
- Publication Status: ☒ Published
- Published On: 2019-02-28 11:50
- Published By: stage2_pod1
- Applies To: * Any *

Endpoints Section:

- Service URL: <https://qa-pod1.staging2.infaqa.com/active-bpel/rt/TaskflowMultiTasks> [View Swagger File](#) [Copy](#)
- SOAP Service URL: <https://qa-pod1.staging2.infaqa.com/active-bpel/soap/TaskflowMultiTasks> [View WSDL File](#) [Copy](#)

At the bottom left, there is a help icon (question mark). At the bottom right, there is a "Close" button.

サービス URL は次の形式を使用します。

<Informatica Intelligent Cloud Services URL>/active-bpel/rt/<API 名>

SOAP サービス URL は次の形式を使用します。

<Informatica Intelligent Cloud Services URL>/active-bpel/soap/<API 名>

関連付けられた Swagger ファイルと WSDL ファイルも表示できます。

3. タスクフローを実行するには、使用するクライアントタイプに基づいて次のいずれかの手順を実行します。

- REST クライアントを使用する場合、Swagger ファイルで使用可能な、サービス URL および API 定義を使用して、REST クライアント経由で要求を送信します。
- SOAP クライアントを使用する場合、WSDL ファイルで使用可能な、SOAP サービス URL および API 定義を使用して、SOAP クライアント経由で要求を送信します。

サービス URL を使用して、ブラウザまたはサードパーティツール経由で入力を渡すことができます。詳細については、「[ブラウザ経由での入力の受け渡し](#)」(ページ 111)または「[REST クライアント経由での入力の受け渡し](#)」(ページ 111)を参照してください。

タスクフロー実行 ID を応答として受信します。

4. 実行 ID を含むタスクフローを監視するには、次のステップのいずれかを実行します。

- 実行 ID を使用し、**[マイジョブ]** ページでタスクフローの実行を監視します。インスタンス名は次の形式を使用します。
<タスク名>-<実行 ID>
- ステータスリソースを使用して、実行 ID に基づくタスクフローのステータスをクエリします。ステータスリソースの使用方法的詳細については、「[ステータスリソースを使用したタスクフローステータスの監視](#)」(ページ 119)を参照してください。

ブラウザ経由での入力の受け渡し

タスクフローを API として実行すると、エンドポイント URL を使用してブラウザ経由で入力を渡すことができます。

1. タスクフローを開き、**[アクション]** > **[プロパティの詳細]** > **[サービス URL のコピー]** をクリックします。
2. 以下に示すように、テキストエディタを開き、入力フィールドと値をサービス URL に追加します。

<Informatica Intelligent Cloud Service URL>/active-bpel/rt/<API_name>?<inputfield>=<value>

例: <https://na1.dm-us.informaticacloud.com/active-bpel/rt/Taskflow?CustomerName=TestConsumer>

複数のフィールドに値を渡すには、以下に示すように&を使用して入力フィールドを区切ります。

[https://na1.dm-us.informaticacloud.com/active-bpel/rt/Taskflow?](https://na1.dm-us.informaticacloud.com/active-bpel/rt/Taskflow?CustomerName=TestConsumer&CustomerEmail=testconsumer@mailinator.com&ItemName=item1&ItemCount=2)

[CustomerName=TestConsumer&CustomerEmail=testconsumer@mailinator.com&ItemName=item1&ItemCount=2](https://na1.dm-us.informaticacloud.com/active-bpel/rt/Taskflow?CustomerName=TestConsumer&CustomerEmail=testconsumer@mailinator.com&ItemName=item1&ItemCount=2)

3. ブラウザを開き、入力フィールドと値を含むサービス URL を貼り付けます。タスクフローで認証を使用する場合は、許可されているユーザー名とパスワードを入力する必要があります。

サービスは、タスクフローの実行 ID を応答として返します。この実行 ID を使用して、タスクフローの実行を監視できます。

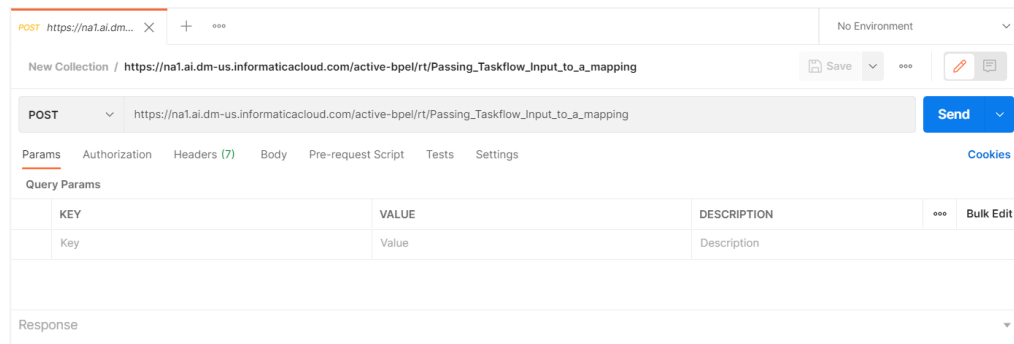
REST クライアント経由での入力の受け渡し

渡す入力値が複数あり、本文が複雑な場合は、Postman などの REST クライアントを使用できます。REST クライアント経由で要求を送信するには、Swagger ファイルで利用可能なサービス URL と API 定義を使用する必要があります。Swagger ファイルには、操作名、認証方法、およびタスクフローの入力が含まれています。Swagger エディタを使用すると、Swagger ファイルの解析や Swagger 要求の取得ができます。

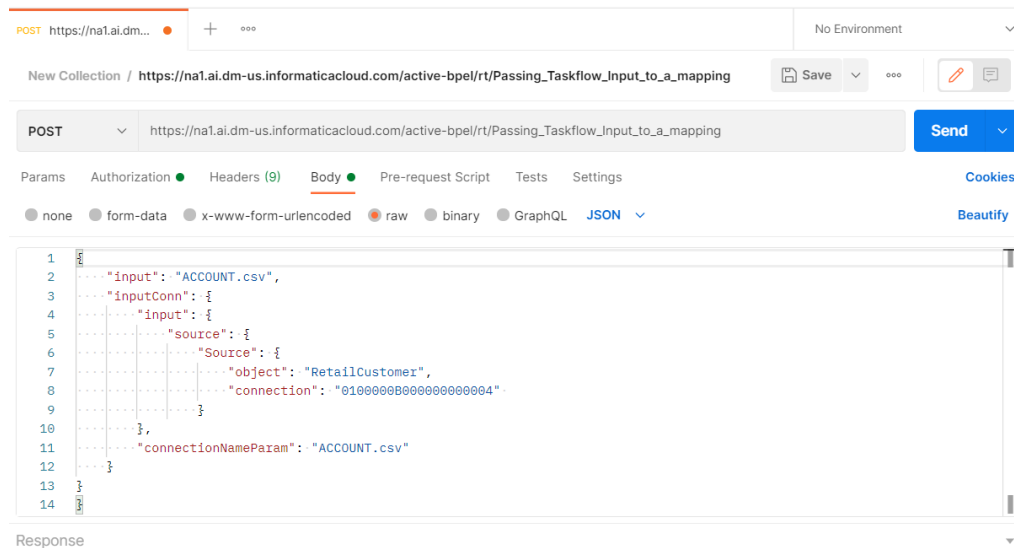
次にサンプルの Swagger 要求を示します。

```
{
  "input": "ACCOUNT.csv",
  "inputConn": {
    "input": {
      "source": {
        "Source": {
          "object": "RetailCustomer",
          "connection": "0100000B000000000004"
        }
      }
    },
    "connectionNameParam": "ACCOUNT.csv"
  }
}
```

1. Postman を開きます。
2. 次の図に示すように、GET や POST などの HTTP 動詞を選択し、生成された REST サービス URL を指定します。



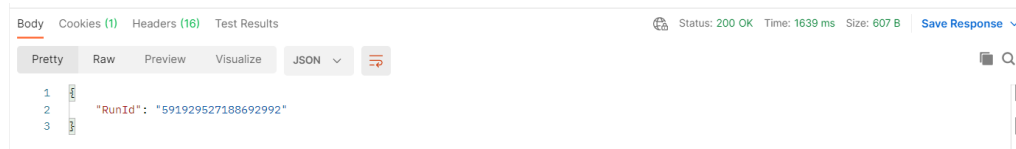
3. 【認証】タブでユーザーアカウントの詳細を入力します。
4. 次の図に示すように、【本文】タブで Swagger 要求を指定します。



5. 要求の本文で入力値を指定します。

6. 【送信】をクリックします。

次の図に示すように、応答としてタスクフロー実行 ID を受信します。



一時停止したタスクフローの再開

resumeWithFaultRetry リソースを使用して、障害が発生したステップから、一時停止したタスクフローインスタンスを再開することができます。また、resumeWithFaultSkip リソースを使用して、障害が発生したステップをスキップし、一時停止したタスクフローインスタンスを次のステップから再開することもできます。

resumeWithFaultRetry

失敗したステップから、一時停止したタスクフローインスタンスを再開するには、PUT 要求で次の URI を使用します。

PUT <Informatica Intelligent Cloud Services URL>/active-bpel/management/runtime/v1/resumeWithFaultRetry/<実行 ID>

要求に次の情報を含めます。

フィールド	タイプ	必須	説明
実行 ID	String	○	タスクフローの実行 ID。

resumeWithFaultRetry PUT の例

このサンプルを参考にして、一時停止したタスクフローインスタンスを、PUT 要求の失敗したステップから再開します。

PUT https://na1.dm-us.informaticacloud.com/active-bpel/management/runtime/v1/resumeWithFaultRetry/681134580186693632

Accept: application/json

INFA-SESSION-ID: 9KA11tLGqxVcGeu18SQBK3

resumeWithFaultRetry PUT response

要求に成功した場合に 204 の応答コードを返します。

エラーが発生した場合はエラーオブジェクトを返します。

resumeWithFaultSkip

失敗したステップをスキップし、一時停止したタスクフローインスタンスを次のステップから再開するには、PUT 要求で次の URI を使用します。

PUT <Informatica Intelligent Cloud Services URL>/active-bpel/management/runtime/v1/resumeWithFaultSkip/<実行 ID>

要求に次の情報を含めます。

フィールド	タイプ	必須	説明
実行 ID	String	○	タスクフローの実行 ID。

resumeWithFaultSkip PUT の例

このサンプルを参考にして、失敗したステップをスキップし、一時停止したタスクフローインスタンスを、PUT 要求の次のステップから再開します。

PUT <https://na1.dm-us.informaticacloud.com/active-bpel/management/runtime/v1/resumeWithFaultSkip/681134580186693632>

Accept: application/json

INFA-SESSION-ID: 9KA11tLGqxVcGeul8SQBK4

resumeWithFaultSkip PUT の応答

要求に成功した場合に 204 の応答コードを返します。

エラーが発生した場合はエラーオブジェクトを返します。

パラメータセットを使用したタスクフローの実行

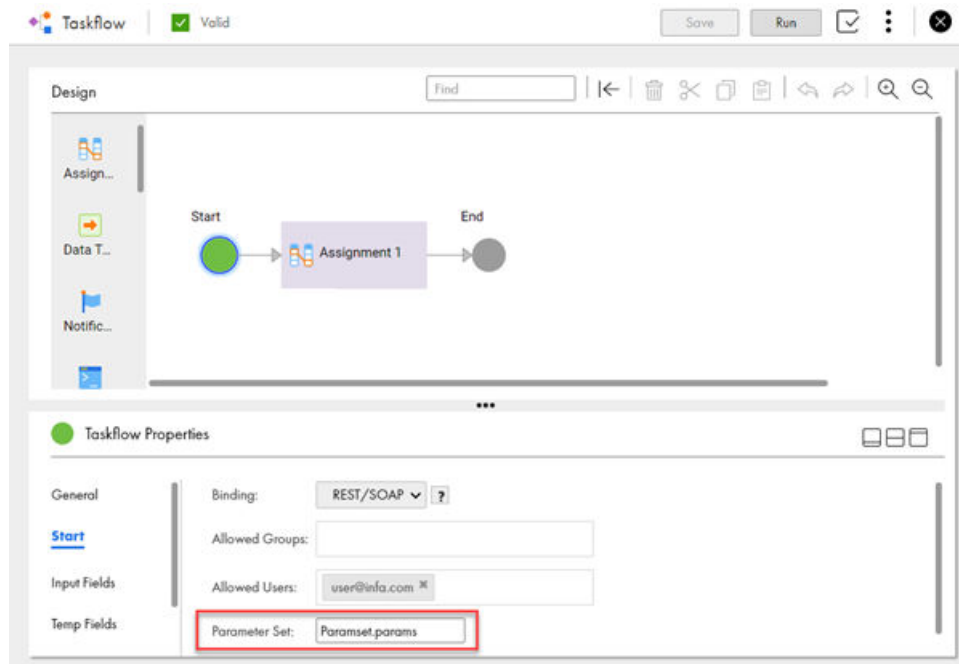
タスクフローの設計中に、クラウドホステッドリポジトリで使用可能なパラメータセット名を指定できます。ただし、パラメータセット名は実行時に上書きできます。

次のビデオは、パラメータセットを使用してタスクフローを設定および実行する方法を示しています。

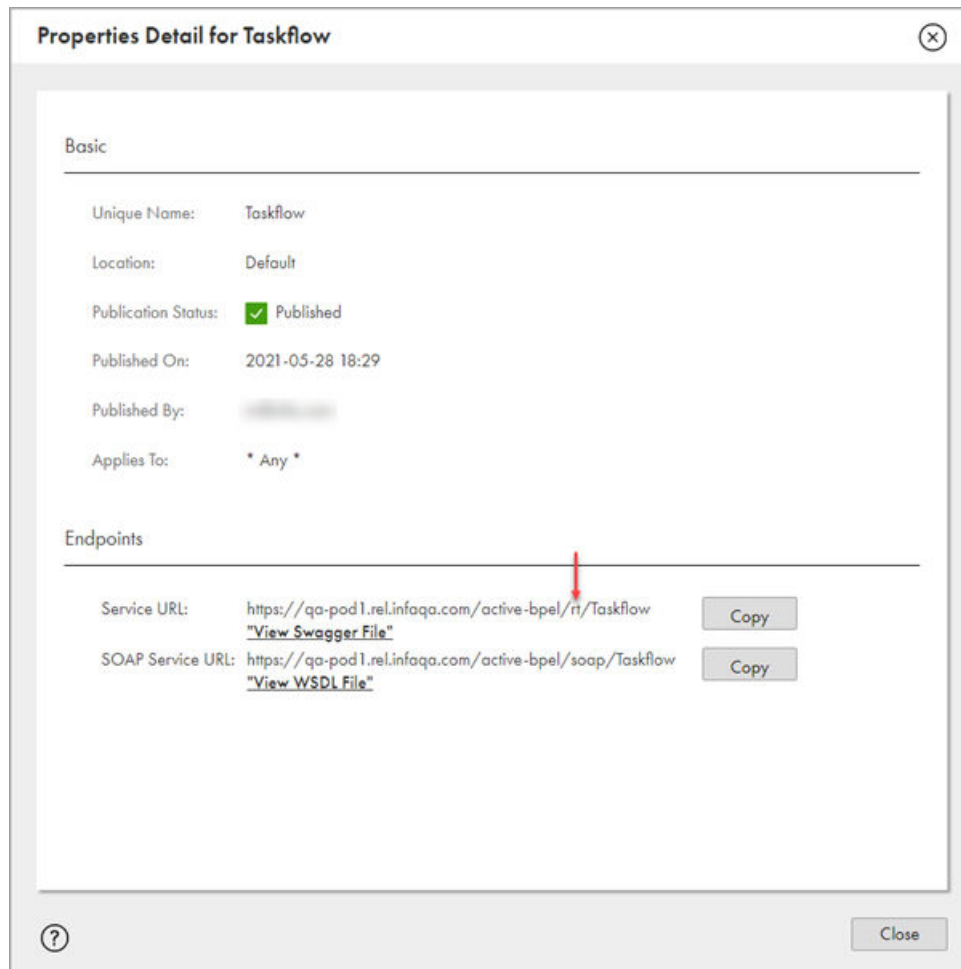
<https://www.youtube.com/watch?v=zDPYS9e0ryM>

タスクフローでパラメータセットを使用するには、次の手順を実行します。

1. データ統合で **【参照】** を選択します。
【参照】 ページが開きます。
2. パラメータセットを割り当てるタスクフローに移動します。
3. **【開始】** ステップを選択し、次の図に示すように、**【開始】** タブの **【パラメータセット】** フィールドにパラメータセット名を入力します。



4. パラメータセットから値を読み取る入力フィールドと一時フィールドを追加します。
5. **【アクション】** メニューで **【パブリッシュ】** を選択し、タスクフローをパブリッシュしてサービス URL を生成します。
6. **【アクション】** メニューで **【プロパティの詳細】** を選択します。
次の図に示すように、**【プロパティの詳細】** ダイアログボックスにサービス URL と SOAP サービス URL が表示されます。



注: パラメータセットを使用してタスクフローを実行する場合に使用できるのはサービス URL のみです。SOAP サービス URL を使用してパラメータセットを実行することはできません。

7. タスクフローを実行するには、Swagger ファイルで利用可能なサービス URL と API 定義を使用して、REST クライアント経由で要求を送信します。

パラメータセットを使用してタスクフローを実行する前に、次に示すように、サービス URL の **rt** を **tf** に変更する必要があります。

変更前: <Informatica Intelligent Cloud Services URL>/active-bpel/**rt**/**<API_name>**

変更後: <Informatica Intelligent Cloud Services URL>/active-bpel/**tf**/**<API_name>**

以下に例を示します。

`https://qa-pod1.rel.infaqa.com/active-bpel/tf/Taskflow`

タスクフロー実行 ID を応答として受信します。

8. パラメータセットを、クラウドホステッドリポジトリで使用可能な別のパラメータセットで実行時に上書きするには、次に示すようにサービス URL を追加します。

以下に例を示します。

<Informatica Intelligent Cloud Services URL>/active-bpel/tf/<API_name>/**paramset**/**<new_parameter_set_name>**

`https://qa-pod1.rel.infaqa.com/active-bpel/tf/Taskflow/paramset/overrideParamset.params`

9. 実行 ID を含むタスクフローを監視するには、次のステップのいずれかを実行します。
- 実行 ID を使用し、[マイジョブ] ページでタスクフローの実行を監視します。インスタンス名は次の形式を使用します。
<タスク名>-<実行 ID>
 - ステータスリソースを使用して、実行 ID に基づくタスクフローのステータスをクエリします。ステータスリソースの使用方法的詳細については、[「ステータスリソースを使用したタスクフローステータスの監視」 \(ページ 119\)](#)を参照してください。

タスクフローの実行 RunAJob ユーティリティの使用

ジョブの実行やジョブのステータスを確認を行う場合は、Informatica Intelligent Cloud Services REST API を介して直接呼び出す代わりに、RunAJob ユーティリティを使用することができます。

RunAJob ユーティリティを使用してタスクフローを実行するには、タスクフローがパブリッシュされている必要があります。また、タスクフローデザイナーの [許可されたユーザー] フィールドおよび [許可されたグループ] フィールドに値を含める必要があります。

RunAJob ユーティリティを使用してタスクフローを実行する方法については、『Rest API リファレンス』を参照してください。

コネクタファイルリスナ経由のタスクフローの呼び出し

コネクタファイルリスナ経由でタスクフローを呼び出す事ができます。

タスクフロー内で、バインディングタイプを [イベント] として定義し、イベントソースとしてコネクタファイルリスナを選択できます。タスクフローをパブリッシュするときに、タスクフローはその中で定義されているコネクタファイルリスナにサブスクライブします。ファイルイベントが発生すると、コネクタファイルリスナはタスクフローを呼び出します。例えば、フォルダ上の新しいファイルをリスンするようにコネクタファイルリスナを設定した場合、コネクタファイルリスナは指定したフォルダに新しいファイルが到着するたびに、関連するタスクフローを呼び出します。

コネクタファイルリスナの実行や、コネクタファイルリスナの各実行ジョブで発生するイベントを監視することができます。Monitor の [ファイル転送ログ] ページには、コネクタファイルリスナのログエントリが一覧表示されます。コネクタファイルリスナのログには、名前、サイズ、最終変更日時、ファイルパス、イベントタイプおよびイベント時間が表示されます。

1. コネクタファイルリスナを作成し、タスクフローを呼び出すイベントを定義します。
2. コネクタファイルリスナを開始またはスケジュールします。
コネクタタスクフローをパブリッシュするときには、ファイルリスナを実行している必要があります。
3. タスクフローを作成します。
4. タスクフローの最初のステップをクリックします。
5. [開始] タブをクリックします。
6. [バインディング] フィールドで [イベント] を選択します。
7. [イベントソース名] フィールドで、設定したコネクタファイルリスナを選択します。

Data Integration は、コネクタファイルリスナイイベントの一部として到着した、更新または削除されたファイルの詳細を格納するための入力フィールドを作成します。入力フィールドはコネクタファイルリシナの名前を取得します。

8. 必要に応じて、タスクフローにステップを追加します。

9. タスクフローを保存してパブリッシュします。

コネクタファイルリシナからサブスクリプト解除するには、タスクフローをパブリッシュ解除します。

タスクフローのスケジュール

タスクフローをスケジュールするには、タスクフローを既存のスケジュールに関連付けるか、新しいスケジュールを作成します。

データ統合と Administrator で新しいスケジュールを作成できます。Administrator でスケジュールを作成する方法に関する詳細については、Administrator のヘルプの「スケジュール」を参照してください。

タスクフローをスケジュールする前に、タスクフローをパブリッシュする必要があります。スケジュール済みのタスクフローは、タスクフローが少なくとも 1 回パブリッシュされている場合にのみ実行されます。

パブリッシュ解除された変更を含む古いタスクフローをスケジュールする場合、データ統合は最後にパブリッシュされたタスクフローバージョンをスケジュールします。

スケジュール済みのタスクフローをパブリッシュ解除すると、スケジュール済みのタスクフロージョブは実行されません。

【アクション】 メニューから **【スケジュール済みのジョブ】** を選択すると、スケジュールされたジョブを確認できます。

注: スケジュール済みのジョブを削除するには、ジョブを選択して **【削除】** をクリックします。

1. データ統合で **【参照】** を選択します。

【参照】 ページが開きます。**【プロジェクトとフォルダ】**、**【アセットタイプ】**、または **【タグ】** オプションを使用して、ページをフィルタリングできます。

2. スケジュールするタスクフローまでナビゲートし、**【アクション】** をクリックします。

【アクション】 メニューが表示されます。

3. **【アクション】** メニューで **【スケジュール】** を選択します。

【スケジュールタスクフロー】 ダイアログボックスが表示されます。

以下の図は、**【スケジュールタスクフロー】** ダイアログボックスを示しています。

Schedule Taskflow

Job Name: * ParallelMTTaskflow

Schedule: * my schedule ▼

Description:

Starts: Jan 2, 2020, 6:05:00 PM India Standard Time

Repeats: Every N Minutes

Next Scheduled Run:

New Schedule Assign Schedule Cancel

4. **[ジョブ名]**フィールドで、このタスクフローとスケジュールの組み合わせに対する名前を入力します。
名前には、英数字、スペース、および次の特殊文字を含めることができます: _ . + -
5. 次のいずれかの手順に従います。
 - 既存のスケジュールを割り当てるには、**[スケジュール]** リストからスケジュールを選択し、**[スケジュールの割り当て]** をクリックします。
 - スケジュールを作成するには、**[新しいスケジュール]** をクリックし、スケジュールの詳細を入力して、**[保存]** をクリックします。作成したスケジュールが **[スケジュール]** リストで選択された状態になります。**[スケジュールの割り当て]** をクリックして、スケジュールをタスクフローに割り当てます。

注: タスクフローをタスクフロー実行中にスケジュールから削除しても、ジョブは完了します。データ統合は、スケジュールに関連付けられた追加の実行を取り消します。

ステータスリソースを使用したタスクフローステータスの監視

Monitor でジョブの結果を表示する特権がある場合は、ステータスリソースを使用してタスクフローのステータスを取得できます。タスクフローの実行 ID をパスパラメータとして使用するか、実行 ID、実行ステータス、開始時刻、終了時刻、オフセット、行制限などのクエリパラメータを使用して、タスクフローのステータスを取得できます。

GET 要求

実行 ID をパスパラメータとして使用してタスクフローのステータスを取得するには、次の URI を使用します。

<Informatica Intelligent Cloud Services URL>/active-bpel/services/tf/status/<run ID>

以下に例を示します。

<https://na4.dm.us.informaticacloud.com/active-bpel/services/tf/status/20262247166322413568>

クエリパラメータを使用して、タスクフローのステータスを取得することもできます。クエリパラメータでは大文字と小文字が区別されます。

GET 要求 URI には次のオプションのクエリパラメータを使用できます。

フィールド	説明
runId	タスクフローの実行 ID。
runStatus	タスクフローの実行ステータス。ステータスは、Success、Failed、Suspended、または Running として指定できます。
startTime	タスクフローの実行が開始した時刻。協定世界時（UTC）を使用します。例: 2021-06-10T05:48:28Z
endTime	タスクフローの実行が終了した時刻。協定世界時（UTC）を使用します。例: 2021-06-11T05:48:28Z
offset	スキップする行数。例えば、最初の 3 行をスキップする場合に使用します。
rowLimit	返す行の最大数。指定可能な最大数値は 50 です。このパラメータを省略すると、クエリは使用可能な行をすべて返します（最大 10 行）。

これらのクエリパラメータは組み合わせて使用することができます。例えば、次のような URI を使用できます。

```
<Informatica Intelligent Cloud Services URL>/active-bpel/services/tf/status?  
runId=<runId>&startTime=<startTime>&runStatus=<runStatus>&endTime=<endTime>&rowLimit=<rowLimit>
```

次のいずれかの方法で GET 要求を認証します。

- 基本認証を使用し、Informatica Intelligent Cloud Services のユーザー名およびパスワードを指定します。以下に例を示します。

```
GET <Informatica Intelligent Cloud Services URL>/active-bpel/services/tf/status/<run ID>  
Accept: application/json  
Authorization: Basic Auth  
username: <Informatica Intelligent Cloud Services user name>  
password: <Informatica Intelligent Cloud Services password>
```

- HTTP ヘッダーの INFA-SESSION-ID を使用します。以下に例を示します。

```
GET <Informatica Intelligent Cloud Services URL>/active-bpel/services/tf/status/<run ID>  
Accept: application/json  
INFA-SESSION-ID: <sessionId>
```

INFA-SESSION-ID を取得するには、Platform REST API バージョン 3 ログインリソースを使用します。ログインリソースの詳細は、『*REST API リファレンス*』を参照してください。

JSON 形式を使用して要求を送信します。ヘッダーに次の行を含めます: Accept: application/json

GET 応答

成功した場合はタスクフローステータス情報が、エラーが発生した場合はエラーオブジェクトが返されます。

成功した場合は、タスクフローの次のステータス情報が返されます。

フィールド	タイプ	説明
assetName	String	タスクフローの名前。
assetType	String	オブジェクトのタイプ。TASKFLOW 値が返されます。

フィールド	タイプ	説明
duration	String	タスクフローが完了、中断、失敗、または停止するまでの実行時間 (秒)。
endTime	Date/Time	タスクフローの実行が終了した時刻。協定世界時 (UTC) を使用します。
ロケーション	String	タスクフローが配置されたプロジェクトおよびフォルダパス。
runId	Long	タスクフローの実行 ID。
runtimeEnv	String	タスクフローが実行するランタイム環境の ID。
runtimeEnvName	String	タスクフローが実行するランタイム環境の名前。
startTime	Date/Time	タスクフローの実行が開始した時刻。協定世界時 (UTC) を使用します。
startedBy	String	タスクフローを開始したユーザー。
ステータス	String	タスクフローの実行ステータス。 タスクフローのステータスを示す、次の値のいずれかを返します。 - RUNNING。タスクフローは実行中です。 - SUCCESS。タスクフローは正常に完了しました。 - FAILED。エラーが発生したため、タスクフローは完了しませんでした。 - SUSPENDED。タスクフローの実行は中断しました。
サブタスク	String	タスクフローに含まれるサブタスクの数。
updateTime	Date/Time	タスクフローの実行ステータスが更新された最終時刻。協定世界時 (UTC) を使用します。
errorMessage	String	エラーメッセージの文字列。
subtaskDetails	String	タスクフロー内のすべてのサブタスクのステータスの詳細を含むオブジェクト。
詳細	String	ステータスの詳細。タスクオブジェクト内の各サブタスクのステータス情報を含みます。
タスク	Collection	タスクフローが含むすべてのサブタスクのステータス情報。

タスクオブジェクトには、タスクフローが含む各サブタスクの次のステータス情報が含まれます。

フィールド	タイプ	説明
assetName	String	タスクフロー内のサブタスクの名前。
assetType	String	サブタスクのタイプ。次のいずれかの値を返します。 - MTT。マッピングタスク。 - DSS。同期タスク。
duration	String	サブタスクが完了、失敗、または停止するまでの実行時間 (秒)。
endTime	Date/Time	サブタスクの実行が終了した時刻。協定世界時 (UTC) を使用します。

フィールド	タイプ	説明
errorMessage	String	エラーメッセージの文字列。
errorRows	String	サブタスクでエラーが発生した行の合計数。
ロケーション	String	サブタスクが配置されたプロジェクトおよびフォルダパス。
rowsProcessed	String	サブタスクで処理された行の合計数。
runId	Long	サブタスクの実行 ID。
runtimeEnv	String	サブタスクが実行するランタイム環境の ID。
runtimeEnvName	String	サブタスクが実行するランタイム環境の名前。
startTime	Date/ Time	サブタスクの実行が開始した時刻。協定世界時（UTC）を使用します。
startedBy	String	タスクを開始したユーザー。このフィールドはタスクフローを開始したユーザーと同じです。
ステータス	String	<p>サブタスクの実行ステータス。</p> <p>サブタスクのステータスを示す、次の値のいずれかを返します。</p> <ul style="list-style-type: none"> - QUEUED。サブタスクは Secure Agent でキューに追加されますが、開始されることはありません。 - STARTING。サブタスクは開始中です。 - RUNNING。サブタスクは実行中です。 - COMPLETED。サブタスクが正常に完了しました。 - STOPPED。タスクフローは実行を停止しているため、サブタスクを開始出来ません。 - WARNING。サブタスクがエラーで完了しました。 - FAILED。エラーが発生したため、サブタスクは完了しませんでした。
サブタスク	String	将来の使用のために予約済み。このフィールドがサブタスクに返されるとき、値は常に 0 です。
successRows	String	サブタスクで正常に処理された行の合計数。
updateTime	Date/ Time	サブタスクの実行ステータスが更新された最終時刻。協定世界時（UTC）を使用します。

次のいずれかの応答を受け取る場合があります。

応答	説明
消去されたログ	インスタンスのログが消去されている場合、応答は次のようになります。 { "status": "No status available." } HTTP ステータスコードは 200 OK です。
無効な実行 ID	実行 ID が無効な場合、応答は次のようになります。 { "error": "CMN_003-Bad request. Error message - The property '<runID>', used in a query expression, is not defined in type 'OData.job-log-service.JobLogEntry'." } HTTP ステータスコードは 400 Bad Request(From JLS)です。
無効なフィルタ句	パラメータ ID が無効な場合、応答は次のようになります。 { "error": "JLS_007-Invalid Filter clause in OData request. RawURI = http://internal-nal-elb.infacloudops.net:443/jls-di/internal/api/v1" } HTTP ステータスコードは 400 Bad Request(From JLS)です。
使用できない JLS サービス	JLS サービスが使用できない場合、応答は次のようになります。 { "error": "503-Service Unavailable." } HTTP ステータスコードは 503 Service Unavailable です。

実行 ID をパスパラメータとして使用する GET の例

次の例は、実行 ID をパスパラメータとして使用するタスクフローステータス要求を示しています。

```
GET https://pod.ics.dev:444/active-bpel/services/tf/status/20262247166322413568
Accept: application/json
INFA-SESSION-ID: 9KA11tLGqxVcGeul8SQBK3
```

タスクフローの設定および要求の入力により、応答は次のタイプになります。

サブタスクなしのタスクフロー

要求が成功し、タスクフローにサブタスクが含まれない場合、次の例に示すように、応答にはタスクフローのステータス情報が含まれます。

```
{
  "assetName": "Taskflow1",
  "assetType": "TASKFLOW",
  "duration": "2",
  "endTime": "2018-12-25T15:56:39Z",
  "location": "Default",
  "runId": "262247166322413568",
  "runtimeEnv": "tf_runtime",
  "runtimeEnvName": "",
  "startTime": "2018-12-25T15:56:37Z",
  "startedBy": "sb",
  "status": "SUCCESS",
  "subtasks": "0",
  "updateTime": "2018-12-25T15:56:39Z",
  "errorMessage": {},
  "subtaskDetails": {
    "details": {}
  }
}
```

HTTP ステータスコードは **200 OK** です。

サブタスクありのタスクフロー

要求が成功し、タスクフローに複数のサブタスクが含まれる場合、次の例に示すように、応答にはタスクフローに含まれる各サブタスクのステータス情報が含まれます。

```
{
  "assetName": "Taskflow2",
  "assetType": "TASKFLOW",
  "duration": "89",
  "endTime": "2018-12-23T17:25:16Z",
  "location": "Default",
  "runId": 20262247166322413568,
  "runtimeEnv": "tf_runtime",
  "runtimeEnvName": "",
  "startTime": "2018-12-23T17:23:47Z",
  "startedBy": "sb",
  "status": "SUCCESS",
  "subtasks": "2",
  "updateTime": "2018-12-23T17:25:17Z",
  "errorMessage": {},
  "subtaskDetails": {
    "details": {
      "tasks": [
        {
          "assetName": "MTR",
          "assetType": "MTT",
          "duration": "3",
          "endTime": "2018-12-23T17:24:45Z",
          "errorMessage": "",
          "errorRows": "0",
          "location": "Default",
          "rowsProcessed": "7",
          "runId": "4",
          "runtimeEnv": "01001Q2500000000000002",
          "runtimeEnvName": "tf_runtime_devagent",
          "startTime": "2018-12-23T17:24:42Z",
          "startedBy": "sb",
          "status": "COMPLETED",
          "subtasks": "0",
          "successRows": "7",
          "updateTime": "2018-12-23T17:24:46Z"
        },
        {
          "assetName": "MTR",
          "assetType": "MTT",
          "duration": "10",
          "endTime": "2018-12-23T17:23:59Z",
          "errorMessage": "",
          "errorRows": "0",
          "location": "Default",
          "rowsProcessed": "7",
          "runId": "3",
          "runtimeEnv": "01001Q2500000000000002",
          "runtimeEnvName": "tf_runtime_devagent",
          "startTime": "2018-12-23T17:23:49Z",
          "startedBy": "sb",
          "status": "COMPLETED",
          "subtasks": "0",
          "successRows": "7",
          "updateTime": "2018-12-23T17:24:00Z"
        }
      ]
    }
  }
}
```

HTTP ステータスコードは **200 OK** です。

クエリパラメータを使用した GET の例

次の例は、クエリパラメータとして実行 ID、実行ステータス、および行制限を使用するタスクフローステータス要求を示しています。

```
GET https://pod.ics.dev:444/active-bpel/services/tf/status?
runId=20262247166322413568&runStatus=Success&rowLimit=3
Accept: application/json
INFA-SESSION-ID: 9KA11tLGqxVcGeul8SQBK3
```

要求が成功した場合、応答の形式はパスパラメータを使用した場合と同じですが、角括弧[]で囲まれます。

要求が成功し、タスクフローにサブタスクが含まれない場合、次の例に示すように、応答にはタスクフローのステータス情報が含まれます。

```
[
  {
    "assetName": "Taskflow1",
    "assetType": "TASKFLOW",
    "duration": "2",
    "endTime": "2018-12-25T15:56:39Z",
    "location": "Default",
    "runId": "262247166322413568",
    "runtimeEnv": "tf_runtime",
    "runtimeEnvName": "",
    "startTime": "2018-12-25T15:56:37Z",
    "startedBy": "sb",
    "status": "SUCCESS",
    "subtasks": "0",
    "updateTime": "2018-12-25T15:56:39Z",
    "errorMessage": {},
    "subtaskDetails": {
      "details": {
        "tasks": []
      }
    }
  }
]
```

HTTP ステータスコードは **200 OK** です。

パラメータなしの GET の例

次の例は、パスパラメータまたはクエリパラメータのないタスクフローステータス要求を示しています。

```
<Informatica Intelligent Cloud Services URL>/active-bpel/services/tf/status
```

応答には、過去 24 時間に実行された最後の 10 個のタスクフローのステータス情報が含まれています。

実行中のタスクフローの GET の例

status リソースを使用して実行中のタスクフローのステータスを取得する場合、応答には、次の例に示すように引用符なしで endTime を null としたタスクフローのステータス情報が含まれます。

```
{
  "assetName": "waitStatus",
  "assetType": "TASKFLOW",
  "duration": 27,
  "endTime": null,
  "errorMessage": "",
  "location": "Default",
  "runId": "737194191850250240",
  "runtimeEnv": "taskflow-preview-usw1-r40-app02.infacloudops.net:4430",
  "runtimeEnvName": "",
  "startedBy": "sb",
  "startTime": "2022-07-28T06:26:32Z",
  "status": "RUNNING",
  "subtasks": 0,
  "updateTime": "2022-07-28T06:26:32Z",
  "subtaskDetails": {
```

```

    "details": {
      "tasks": []
    }
  }
}

```

タスクフローが完了したら、正しい endTime 値が表示されます。

タスクフローの例

複数のデータ統合タスクをタスクフローに追加して、それらをシーケンシャルまたは並列に実行できます。

次の例は、並列に 2 つのマッピングタスクを実行するタスクフローの作成方法を示しています。この例は、接続、マッピング、およびマッピングタスクの作成方法を読者が理解していることを前提としています。


この例では、次のサンプルマッピングタスクを使用します。

- フラットファイルデータベースをフラットファイルデータベースにマップするマッピングタスク。
- MySQL データベースをフラットファイルデータベースにマップするマッピングタスク。

次のステップを実行してタスクフローを作成し、タスクフローの進捗を監視します。

1. トップメニューバーから、データ統合を選択します。
2. **【新規作成】** > **【タスクフロー】** > **【タスクフロー】** > **【作成】** をクリックします。
3. **【開始】** を選択して、次の全般プロパティを入力します。

プロパティ	値
名前	Taskflow2MT
場所	【選択】 をクリックして、タスクフローを保存する場所を参照します。
説明	2 つのマッピングタスクを並列に実行するタスクフロー

 Taskflow2MT Properties

General

Input Fields

Temp Fields

Notes

Step Type: Start
Name: *
Location:
Description:

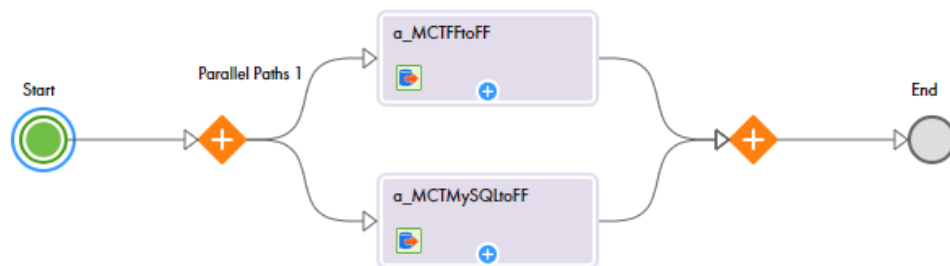
このユースケースでは、どの **【入力フィールド】**、**【一時フィールド】**または **【注意事項】** プロパティも入力する必要はありません。

4. **【並列パス】** ステップをキャンバス上にドラッグします。
【並列パス】 ステップには 2 つのブランチがあります。

5. データタスクステップを追加するには、データタスクステップを左側パレットから並列パスステップの各ブランチにドラッグします。
6. 最初のデータタスクステップを選択し、次のステップを実行します。
 - a. **【プロパティ】 > 【データタスク】** に移動し、**【選択】** をクリックします。マッピングタスクのリストがあるダイアログボックスを表示します。最初のデータタスクステップで使用するタスクを選択し、**【選択】** をクリックします。
 - b. **【プロパティ】 > 【全般】** と移動し、**【名前】** フィールドにデータタスクステップの名前を入力します。
7. 2 番目の並列パスブランチで前述のデータタスク用ステップを繰り返します。
8. 並列パスステップを選択し、**【プロパティ】 > 【一般】** に移動し、**【タイトル】** フィールドにステップの名前を入力します。
9. **【並列パスのプロパティ】 > 【並列パス】** に移動します。タスクフローで並列パスステップのパス 0 とパス 1 が設定されていることを確認します。



10. ページの右上隅から **【保存】** をクリックします。
次の図は、完全なタスクフローを示しています。



11. 検証エラーが表示されない場合は、**【実行】** をクリックします。
注: 検証エラーが表示される場合には、**【検証】** パネルを使用してタスクフローをトラブルシューティングします。
12. タスクフローが完了したことを確認するために、**【監視】** のモニターページまたは **【データ統合】** のデータ統合ページを確認します。

第 3 章

リニアタスクフロー

リニアタスクフローを作成して、複数のデータ統合タスクをグループ化できます。リニアタスクフローは指定した順序でタスクを順番に実行します。

単純なタスクフローを作成するには、リニアタスクフローを使用します。例えば、連絡先のリストを毎月更新するなどです。そうするには、最新のアカウント情報を更新/挿入し、次に各アカウントの連絡先情報を更新/挿入します。タスクフローを、アカウントを更新/挿入する同期タスク、続いてアカウントの契約を更新/挿入する同期タスクで作成します。各月に実行するリニアタスクフローをスケジュールします。

Linear Taskflow Details

Linear Taskflow Name:

Location:

Description:

Schedule: ☒ Do not run this task on a schedule ☐ Run this task on a schedule:

List of Tasks

Sequence	Name	Type	Stop on Error	Stop on Warning	Delete
1	synch_AccountInfo	Synchronization	<input type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="Delete"/>
2	synch_ContactInfo	Synchronization	<input type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="Delete"/>

Email Notification Options

☒ Use the default email notification options for my organization

☐ Use custom email notification options for this task:

Failure Email Notification:

Warning Email Notification:

リニアタスクフローは、データ統合タスクフロー機能の簡略バージョンです。リニアタスクフローは、タスクフロー内の前のタスクに基づいてタスクの実行シーケンスを制御できません。動的タスクフローを作成するには、[第 2 章, 「タスクフロー」 \(ページ 9\)](#)を参照してください。

リニアタスクフローには次のタスクタイプを含めることができます。

- 同期
- レプリケーション
- マッピング
- マスキングタスク
- PowerCenter task: PowerCenter タスク

リニアタスクフローを編集できます。現在実行中のリニアタスクフローにタスクを追加すると、データ統合はリニアタスクフローを次回実行するまで新しいタスクを実行しません。

注: 従来のリニアタスクフローを削除する場合は、組織内にそのタスクを使用する予定のユーザーがいないことを確認してください。削除した後、リニアタスクフローを取得することはできません。

リニアタスクフロージョブのスケジュール設定

リニアタスクフローは、手動で実行することも、スケジュールを使用して、時間単位、日次、週単位などの特定の時間または間隔で実行することもできます。

スケジュールに基づいてリニアタスクフローを実行するには、構成時にリニアタスクフローをスケジュールと関連付けます。既存のスケジュールを使用することも、新規のスケジュールを作成することもできます。

リニアタスクフローの実行中にこれをスケジュールから削除しても、ジョブは完了します。データ統合は、スケジュールに関連付けられた追加の実行を取り消します。

スケジュールの使用と作成の詳細については、『タスク』を参照してください。

リニアタスクフローの設定

複数のタスクをリニアタスクフローに追加できます。ただし、リニアタスクフローには特定のタスクの複数のインスタンスを含めることはできません。タスクフロータスクをリニアタスクフローに追加できます。

1. リニアタスクフローを作成するには、**[新規]** > **[タスクフロー]** > **[リニアタスクフロー]** をクリックし、次に **[作成]** をクリックします。

リニアタスクフローを編集するには、**[エクスプローラ]** ページで、リニアタスクフローにナビゲートします。リニアタスクフローが含まれている行で、**[アクション]** をクリックし、**[編集]** を選択します。

2. リニアタスクフローの次の詳細を入力します。

フィールド	説明
タスクフロー名	リニアタスクフローの名前。名前には、英数字、スペース、および次の特殊文字を含めることができます。 _ . + - 名前の大文字と小文字は区別されません。
場所	リニアタスクフローの場所。リニアタスクフローを保存するフォルダを参照するか、またはデフォルトの場所を使用します。 [Explore (参照)] ページが現在アクティブになっていて、プロジェクトまたはフォルダが選択されている場合、アセットのデフォルトの場所はその選択されているプロジェクトまたはフォルダです。そうでない場合、デフォルトの場所は直近で保存されたアセットの場所です。

フィールド	説明
説明	リニアタスクフローの説明。
スケジュール	<p>タスクの実行方法を決定します。次のオプションから選択します。</p> <ul style="list-style-type: none"> - 手動。リニアタスクフローとタスクが正しく設定されていることを確認するには、リニアタスクフローを手動で実行できます。 タスクを手動で実行するには、【このタスクはスケジュールを使用しない】 をクリックします。 - スケジュール。リニアタスクフローがスケジュールに従って実行されるように設定する場合、リニアタスクフローを定期的に実行する頻度を設定します。 タスクをスケジュールに関連付けるには、【このタスクは指定したスケジュールを使用する】 をクリックしてスケジュールを選択します。 スケジュールを作成するには、【新規】 をクリックします。

3. タスクをリニアタスクフローに追加するには、以下の手順を実行します。
 - a. **【タスクの追加】** をクリックします。
 - b. タスクがあるフォルダにナビゲートします。
 - c. リニアタスクフローに含めるタスクを選択し、**【選択】** をクリックします。
 - d. これらのステップを繰り返して、要求どおりにタスクを追加します。
4. タスクが実行する順序を決定するには、タスクリスト内の各タスクのシーケンス番号を入力します。
5. タスクでエラーが発生した場合にリニアタスクフローが停止するようにするには、そのタスクで**【エラー時に停止】** をクリックします。
 選択したタスクのいずれかでエラーが発生した場合、エージェントは残るすべてのタスクの実行を停止します。リニアタスクフローの実行中にこのオプションを変更した場合、データ統合は次回リニアタスクフローを実行するまで変更を適用しません。
6. リニアタスクフローからタスクを削除するには、タスクの横にある**【削除】** アイコンをクリックします。
7. オプションで、次の電子メール通知オプションを設定します。

フィールド	説明
組織でデフォルトの電子メール通知オプションを使用する	組織に設定されている電子メール通知オプションを使用します。
このタスク用のカスタム電子メール通知オプションを使用	<p>タスクに設定されている電子メール通知オプションを使用します。タスクが失敗した場合、エラーが発生して終了した場合、正常に終了した場合に基づいて、それぞれのアドレスに電子メールを送信できます。</p> <p>電子メールアドレスのリストを区切るには、カンマを使用します。</p> <p>このオプションを選択すると、Data Integration は、組織に設定した電子メール通知のオプションを使用しません。</p>

8. リニアタスクフローを保存するには、**【保存】** をクリックします。
9. リニアタスクフローをテストするには、**【実行】** をクリックして **【マイジョブ】** ページで結果を確認します。

リニアタスクフローの実行

シーケンスのタスクのリストを実行するには、リニアタスクフローを使用します。

リニアタスクフローは、次の方法で実行できます。

- 手動
リニアタスクフローを手動で実行するには、**【エクスプローラ】** ページでタスクフローに移動します。タスクフローが含まれている行で、**【アクション】** をクリックし、**【実行】** を選択します。または、リニアタスクフローを開いて、**【実行】** をクリックできます。
- スケジュール
リニアタスクフローをスケジュールに基づいて実行するには、タスクフローを設定してそれをスケジュールに関連付けます。

リニアタスクフローまたはサブタスクの停止

【マイジョブ】 ページでリニアタスクフロージョブを停止することができます。

リニアタスクフローで実行中のタスクを停止することもできます。タスクが予想より長時間実行されているときに、場合によっては、リニアタスクフローで実行中のタスクを停止する必要があります。タスク内で実行中のサブタスクは停止できません。

最後のタスクを停止すると、リニアタスクフロージョブのステータスは**【失敗】** になります。

リニアタスクフローの最後のタスク以外のタスクを停止する場合、リニアタスクフロージョブが停止するか続行するかは、**【エラー時に停止】** プロパティが有効になっているかどうかによって決まります。

- **【エラー時に停止】** を有効にすると、リニアタスクフロージョブの実行は停止します。ジョブのステータスは**【失敗】** になります。
- **【エラー時に停止】** が有効になっていない場合、リニアタスクフローはフロー内の次のタスクで再開します。ジョブが完了すると、ジョブのステータスは**【警告】** になります。これは、リニアタスクフローのすべてのタスクが完了しなかったためです。

タスクとタスクフロージョブの停止の詳細については、『**タスク**』を参照してください。ジョブの監視と停止の詳細については、Monitor のヘルプを参照してください。

索引

数字

10 進数値
変換 [96](#)

A

addToDate 関数
説明 [66](#)

C

Cloud Application Integration コミュニティ
URL [6](#)
Cloud 開発者コミュニティ
URL [6](#)

D

dateDiff 関数
説明 [69](#)
decode 関数
説明 [72](#)

G

getAssetLocation 関数
説明 [75](#)
getAssetName 関数
説明 [75](#)
getDatePart 関数
説明 [75](#)
getInstanceStartTime 関数
説明 [77](#)

I

IIF 関数
説明 [78](#)
Informatica Intelligent Cloud Services
Web サイト [6](#)
Informatica グローバルカスタマサポート
連絡先情報 [7](#)
instr 関数
説明 [80](#)
in 関数
説明 [79](#)
isNull 関数
説明 [83](#)

L

lastDay 関数
説明 [84](#)
lpad 関数
説明 [85](#)
ltrim 関数
説明 [87](#)

M

month
最終日を返す [84](#)

N

NULL 値
isNull [83](#)
確認の対象 [83](#)

R

round (Numbers)関数
説明 [88](#)
rtrim 関数
説明 [90](#)

T

toChar (Numbers)関数
説明 [92](#)
toDate 関数
説明 [94](#)
toDecimal 関数
説明 [96](#)
toInteger 関数
説明 [98](#)
trunc 関数
サブ秒の処理 [99](#)
説明 [99](#)

W

Web サイト [6](#)

あ

アップグレード通知 [7](#)

こ

コンポーネント
リニアタスクフロー [128](#)

さ

サブ秒
trunc 関数での処理 [99](#)

し

システムステータス [7](#)
ジョブ
リニアタスクフローのスケジュール設定 [129](#)

す

スケジュール設定
リニアタスクフロー [129](#)
ステータス
Informatica Intelligent Cloud Services [7](#)
ステータスリソース
タスクフロー [119](#)

た

タスクフロー
[検証] パネル [102](#)
タイプ [8](#)
タスクフローステップのプロパティ [22](#)
タスクフローテンプレート [11](#)
タスクフローのステップ [9](#)
タスクフローのパラメータ [53](#)
タスクフローの作成 [11](#)
タスクフローの実行 [103](#)
タスクフローの例 [126](#)

タスクフロー (続く)
タスクフロープロパティ [13](#)
概要 [9](#)
式エディタ [62](#)
タスクフロー、リニア
設定 [129](#)
タスクフロー関数
getAssetLocation [75](#)
getAssetName [75](#)
getInstanceStartTime [77](#)

て

データクレンジング関数
in [79](#)
テキスト文字列
数値の変換 [92](#)
テスト関数
isNull [83](#)

ふ

フォーマット
文字文字列から日付 [94](#)

め

メンテナンスの停止 [7](#)

り

リニアタスクフロー
スケジュール設定 [129](#)
実行 [131](#)
設定 [129](#)
説明 [128](#)
停止 [131](#)