HITACHI
Inspire the Next

Hitachi Ops Center Automator

Service Builder ユーザーズガイド

4010-1J-037-70

対象製品
Hitachi Ops Center Automator 10.9.3

輸出管理に関する注意
本マニュアル固有の技術データおよび技術は、米国輸出管理法、および関連の規制を含む米国の輸出管理法の対象となる場合があり、その他の国の輸出または輸入規制の対象となる場合もあります。読者は、かかるすべての規制を厳守することに同意し、マニュアルおよび該当製品の輸出、再輸出、または輸入許可を取得する責任があることを了解するものとします。

商標類
HITACHI は，株式会社 日立製作所の商標または登録商標です。
Active Directory は，マイクロソフト 企業グループの商標です。
Linux は，Linus Torvalds 氏の米国およびその他の国における登録商標です。
Microsoft は，マイクロソフト 企業グループの商標です。
Red Hat is a registered trademark of Red Hat, Inc. in the United States and other countries.
Red Hat は，米国およびその他の国における Red Hat, Inc.の登録商標です。
Red Hat Enterprise Linux is a registered trademark of Red Hat, Inc. in the United States and other countries.
Red Hat Enterprise Linux は，米国およびその他の国における Red Hat, Inc.の登録商標です。
ServiceNow，ServiceNow のロゴ，Now，その他の ServiceNow マークは米国および/またはその他の国における ServiceNow, Inc.の商標または登録商標です。
UNIX は，The Open Group の登録商標です。
Windows は，マイクロソフト 企業グループの商標です。
Windows Server は，マイクロソフト 企業グループの商標です。
This product includes software developed by the Apache Software Foundation (http://www.apache.org/).
Portions of this software were developed at the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign.
This product includes software developed by the University of California, Berkeley and its contributors.
This software contains code derived from the RSA Data Security Inc. MD5 Message-Digest Algorithm, including various modifications by Spyglass Inc., Carnegie Mellon University, and Bell Communications Research, Inc (Bellcore).
Regular expression support is provided by the PCRE library package, which is open source software, written by Philip Hazel, and copyright by the University of Cambridge, England. The original software is available from ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/
1. This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)
2. This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)
3. This product includes software written by Tim Hudson (tjh@cryptsoft.com)
4. This product includes the OpenSSL Toolkit software used under OpenSSL License and Original SSLeay License. OpenSSL License and Original SSLeay License are as follow:
LICENSE ISSUES
==========
The OpenSSL toolkit stays under a double license, i.e. both the conditions of
the OpenSSL License and the original SSLeay license apply to the toolkit.
See below for the actual license texts.
OpenSSL License

/* ==========

* Hudson (tjh@cryptsoft.com).
*
*/
Original SSLeay License

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
* All rights reserved.
*
* This package is an SSL implementation written
* by Eric Young (eay@cryptsoft.com).
* The implementation was written so as to conform with Netscapes SSL.
*
* This library is free for commercial and non-commercial use as long as
* the following conditions are aheared to. The following conditions
* apply to all code found in this distribution, be it the RC4, RSA,
* lhash, DES, etc., code; not just the SSL code. The SSL documentation
* included with this distribution is covered by the same copyright terms
* except that the holder is Tim Hudson (tjh@cryptsoft.com).
*
* Copyright remains Eric Young's, and as such any Copyright notices in
* the code are not to be removed.
* If this package is used in a product, Eric Young should be given attribution
* as the author of the parts of the library used.
* This can be in the form of a textual message at program startup or
* in documentation (online or textual) provided with the package.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the copyright
* notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in the
* documentation and/or other materials provided with the distribution.
* 3. All advertising materials mentioning features or use of this software
* must display the following acknowledgement:
* "This product includes cryptographic software written by
* Eric Young (eay@cryptsoft.com)"
* The word 'cryptographic' can be left out if the rouines from the library
* being used are not cryptographic related :-).
* 4. If you include any Windows specific code (or a derivative thereof) from
* the apps directory (application code) you must include an acknowledgement:
* "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
*
* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL

発行
2023 年 9 月 4010-1J-037-70

目次

はじめに

このマニュアルでは、Ops Center Automator Service Builder の使用方法を説明します。

対象読者

このマニュアルは、Admin ロールおよび Develop ロールにおいて、Hitachi Ops Center Automator を使用するストレージ管理者を対象としています。

Service Builder を使用するには、Hitachi Ops Center Automator の概念、用語、および機能を熟知しておく必要があります。

マニュアルの構成

このマニュアルは、次に示す章と付録から構成されています。

第 1 章　Automator Service Builder の概要

Service Builder の概要について説明しています。

第 2 章　既存のサービステンプレートを操作する

［Service Builder Home］画面から、適切なオプションを選択してサービステンプレートを作成し、管理する方法について説明しています。

第 3 章　既存の部品を操作する

部品の概要および部品を管理する方法について説明しています。

第 4 章　新しいサービステンプレートを作成する

サービステンプレートを作成し編集する方法について説明しています。

第 5 章　新しい部品を作成する

部品を作成し編集する方法について説明しています。

第 6 章　ビルド、デバッグ、およびリリース

サービステンプレートおよび部品をデバッグしリリースする方法について説明しています。

第 7 章　高度なオプション

サービステンプレートと部品の管理に利用可能な他の機能について説明しています。

付録 A　参照情報

ビルトインサービステンプレート、ビルトイン部品、予約プロパティの一覧や、部品用のロケール設定について説明しています。

付録 B　ビルトイン部品の説明

ビルトイン部品について説明しています。

マイクロソフト製品の表記について

このマニュアルでは、マイクロソフト製品の名称を次のように表記しています。

表記	製品名
Windows	次の製品を区別する必要がない場合の表記です。 ・Microsoft ® Windows Server ® 2012 ・Microsoft ® Windows Server ® 2012 R2 ・Microsoft ® Windows Server ® 2016 ・Microsoft ® Windows Server ® 2019 ・Microsoft ® Windows Server ® 2022
Windows Server 2012	次の製品を区別する必要がない場合の表記です。 ・Microsoft ® Windows Server ® 2012 ・Microsoft ® Windows Server ® 2012 R2
Windows Server 2016	Microsoft ® Windows Server ® 2016
Windows Server 2019	Microsoft ® Windows Server ® 2019
Windows Server 2022	Microsoft ® Windows Server ® 2022

関連マニュアル

このマニュアルの関連マニュアルを次に示します。必要に応じてお読みください。

- Hitachi Ops Center Automator ユーザーズガイド, 4010-1J-034
- Hitachi Ops Center Automator インストールガイド, 4010-1J-035
- Hitachi Ops Center Automator メッセージ, 4010-1J-038
- Hitachi Ops Center インストールガイド, 4010-1J-101

このマニュアルで使用している記号

このマニュアルでは、次のような表記規則を使用しています。

規則	説明
太字	リスト項目の中で強調する語を示します。
[]	画面のタイトル、メニュー、メニューオプション、ボタン、フィールド、ラベルなど、画面内のテキストを示します。 例：[OK] をクリックします。
＜ ＞ （山括弧）	可変値を示します。
斜体	
`Monospace`	画面に表示されるテキスト、またはユーザーが入力するテキストを示します。例：`pairdisplay -g oradb`
[ ]（角括弧）	オプションの値を示します。例：[a | b]は、a または b を選択できる、あるいはどちらも省略できることを示します。
{ }（波括弧）	必須の値または予期される値を示します。例：{ a | b }は、a または b のどちらかを選択する必要があることを示します。
|（縦線）	2 つ以上のオプションまたは引数から選択できることを示します。例： [a | b]は、a または b を選択できる、あるいはどちらも省略できることを示します。 { a | b }は、a または b のどちらかを選択する必要があることを示します。

KB（キロバイト）などの単位表記について

1KB（キロバイト）、1MB（メガバイト）、1GB（ギガバイト）、1TB（テラバイト）は、それぞれ1KiB（キビバイト）、1MiB（メビバイト）、1GiB（ギビバイト）、1TiB（テビバイト）と読み替えてください。

1KiB、1MiB、1GiB、1TiB は、それぞれ 1,024 バイト、1,024KiB、1,024MiB、1,024GiB です。

このマニュアルでの表記

このマニュアルでは、製品の名称を省略して表記しています。このマニュアルでの表記と、製品の正式名称または意味を次に示します。

表記	製品名
Common Services	Hitachi Ops Center Common Services
Configuration Manager	Hitachi Ops Center API Configuration Manager
Fabric OS（FOS）	Fabric OS®
HUS VM	Hitachi Unified Storage VM
Linux	Red Hat Enterprise Linux®および Oracle Linux®の総称です。
Ops Center Automator（Automator）	Hitachi Ops Center Automator
Ops Center Portal	Hitachi Ops Center Portal
Ops Center Viewpoint	Hitachi Ops Center Viewpoint
Oracle Linux	Oracle Linux®
Red Hat Enterprise Linux（RHEL）	Red Hat Enterprise Linux®
Service Builder	Ops Center Automator Service Builder
Virtual Storage Platform	Hitachi Virtual Storage Platform
	Hitachi Virtual Storage Platform VP9500
VMware	VMware®
VMware ESX	VMware vSphere® ESXi™
VMware vCenter Server	VMware vCenter Server™
VMware vSphere	VMware vSphere®
VSP 5000 シリーズ	Hitachi Virtual Storage Platform 5100
	Hitachi Virtual Storage Platform 5200
	Hitachi Virtual Storage Platform 5500
	Hitachi Virtual Storage Platform 5600
	Hitachi Virtual Storage Platform 5100H
	Hitachi Virtual Storage Platform 5200H
	Hitachi Virtual Storage Platform 5500H
	Hitachi Virtual Storage Platform 5600H
VSP E シリーズ	Hitachi Virtual Storage Platform E390

表記	製品名
	Hitachi Virtual Storage Platform E590
	Hitachi Virtual Storage Platform E790
	Hitachi Virtual Storage Platform E990
	Hitachi Virtual Storage Platform E1090
	Hitachi Virtual Storage Platform E390H
	Hitachi Virtual Storage Platform E590H
	Hitachi Virtual Storage Platform E790H
	Hitachi Virtual Storage Platform E1090H
VSP Fx00 モデル	Hitachi Virtual Storage Platform F350
	Hitachi Virtual Storage Platform F370
	Hitachi Virtual Storage Platform F400
	Hitachi Virtual Storage Platform F600
	Hitachi Virtual Storage Platform F700
	Hitachi Virtual Storage Platform F800
	Hitachi Virtual Storage Platform F900
VSP F1500	Hitachi Virtual Storage Platform F1500
VSP F シリーズ	VSP Fx00 モデル
	VSP F1500
VSP Gx00 モデル	Hitachi Virtual Storage Platform G100
	Hitachi Virtual Storage Platform G130
	Hitachi Virtual Storage Platform G150
	Hitachi Virtual Storage Platform G200
	Hitachi Virtual Storage Platform G350
	Hitachi Virtual Storage Platform G370
	Hitachi Virtual Storage Platform G400
	Hitachi Virtual Storage Platform G600
	Hitachi Virtual Storage Platform G700
	Hitachi Virtual Storage Platform G800
	Hitachi Virtual Storage Platform G900
VSP G1000	Hitachi Virtual Storage Platform G1000
VSP G1500	Hitachi Virtual Storage Platform G1500
VSP G シリーズ	VSP Gx00 モデル
	VSP G1000
	VSP G1500
VSP ファミリー	Virtual Storage Platform
	VSP 5000 シリーズ
	VSP E シリーズ
	VSP F シリーズ
	VSP G シリーズ

1

Automator Service Builder の概要

Service Builder は、タスクの実行の自動化、ストレージリソースのプロビジョニング/割り当て、および特定のデータセンターの IT プロセスの自動化を行うために必要な動作パラメーターを提供するサービステンプレートの管理と作成のための強力なインタフェースを提供します。

サービステンプレートは、スクリプトの実行、指定したシステムへのコマンドの実行、タスクの実行中に使用される入力および出力プロパティの値の提供のための基本要素として機能する部品に基づいています。サービステンプレートと部品は、メインのサービステンプレート内の特定のタスクのセットに、処理の流れを命令する一連のステップとして結びつけることができます。

Ops Center Automator をインストールすると Service Builder も同時に構成されるため、独立したテスト環境に Ops Center Automator をインストールする必要はありません。サービス、部品、構成ファイル、およびマッピングパラメーターをテストすることで、リリースの前に品質と機能性を確実にできます。

事前設定済みのサービステンプレートと、関連付けられている部品が、そのままもしくは最小限の変更で、通常のデータセンターで必要とされるより一般的なタスクの多くを実行できるよう提供されます。事前に準備されたサービステンプレートと部品がサイトのニーズに不十分である場合には、独自のサービステンプレートと部品をビルドし、実行するタスクのタイプを定義できます。また、動作パラメーター、ユーザー情報、および接続設定の構成方法を指定したり、サービスの実行時にオペレーターが使用するユーザーインタフェースの要素をカスタマイズしたりすることもできます。

必要な部品を持つテンプレートを準備し、入力および出力プロパティの設定を定義したら、Service Builder のガイドに従って、データセンターで使用するためにサービステンプレートを実行、デバッグ、およびリリースする最終段階を実行します。

1.1 Service Builder について

Service Builder を使用して、データセンターのタスクを自動化するためのサービステンプレート、および関連する部品を作成して管理できます。

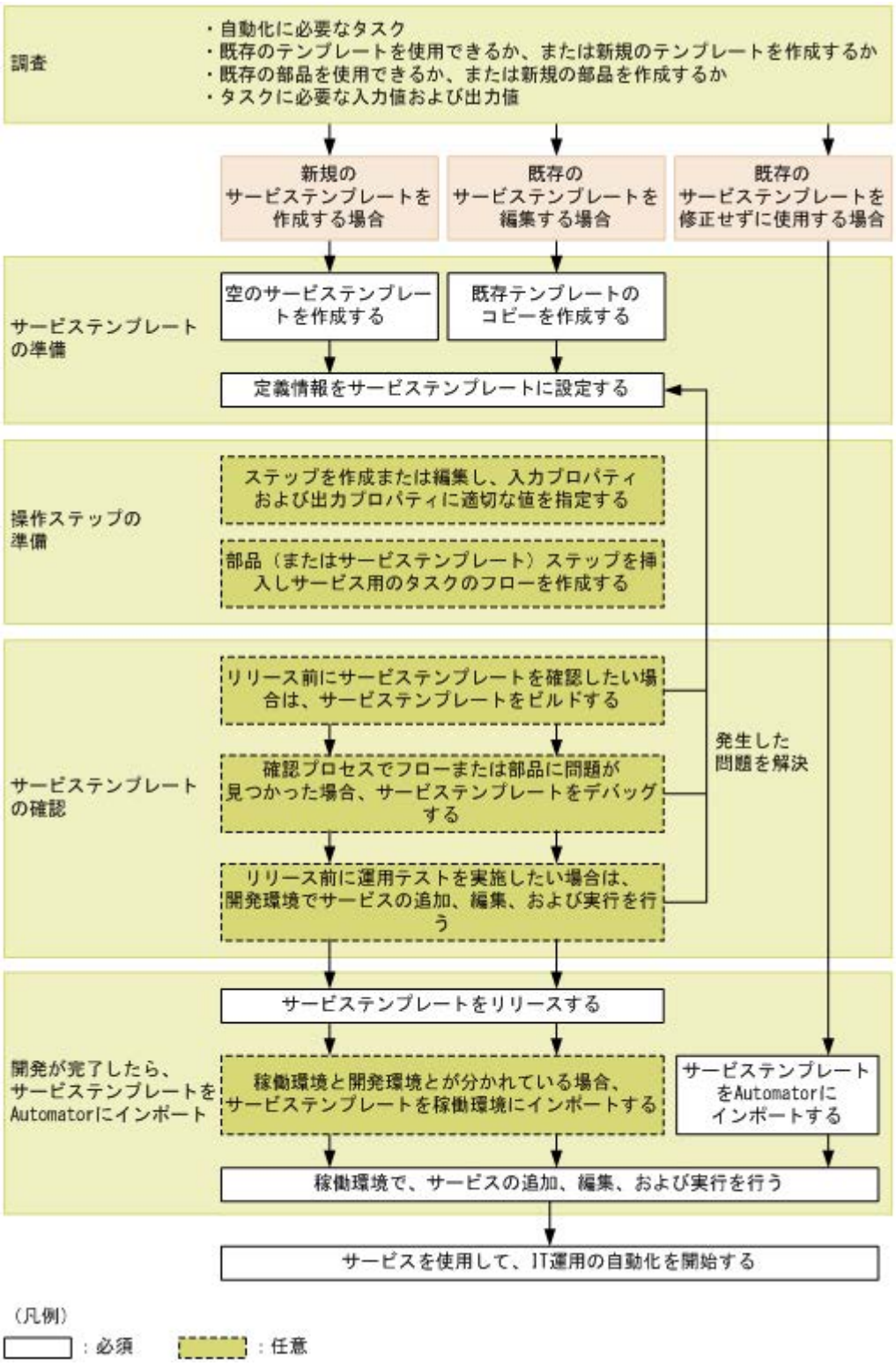
サービステンプレートを使用するとき、パッケージ化されているサービステンプレートの 1 つをそのまま使用するか、要求に応じて既存のテンプレートを変更するか、または使用している環境で要求される特定のタスクを実行する新しいテンプレートを作成するかを選択できます。サービステンプレートはスクリプトを実行する一連の部品で構成され、各種のタスクを達成するために必要なコマンドを実行し、オプションのパラメーターを提供します。

Service Builder は、部品でフローを作成し、それらの要素をカスタマイズしたサービステンプレートにパッケージ化できます。サービステンプレートと部品はどちらもパッケージとカスタマイズ済みで、再利用可能なため、貴重な開発時間と労力を削減できます。

Service Builder では、次の作業が行えます。

- 便利なメニュー操作のインタフェースにより、サービステンプレートと部品を簡単に管理（コピー、編集、削除、更新、エクスポート）できます。
- 提供されているサービステンプレートと部品を使用またはカスタマイズでき、特定のサイト用のタスクを実行するため必要なスクリプトを実行し、コマンドを実行するようカスタマイズされた新しいものを作成することもできます。
- 一連の部品が実行されるステップのフローを、それらの配置に従い、コンポーネント間にコネクターの線を描画するだけで構築できます。
- 特定の部品に関連付けられる入力および出力プロパティの値を指定できます。これらの値は、サービスの実行、必要なエラー処理の実行、およびアラートの生成に必要なホストの詳細、ユーザー情報、接続の設定などを提供します。
- コマンドの実行、スクリプトの実行、およびサービスの実行に変数を使用できます。
- 特定のユーザーグループまたはサービス種別に従ってタスクを分類します。
- 表示されるアイコンやグラフィックの種類、提供される指示の種類、およびサービスをサブミットするために必要な詳細を提供するときユーザーに対して表示される情報をコントロールして、GUI をカスタマイズします。
- サービステンプレートのデバッグとテストを行い、最終的にデータセンターで展開するためリリースします。
- サービステンプレートと設定のインポートとエクスポートを行い、必要に応じて再利用できるようにします。
- ファイルに格納されているプロパティの値のインポートとエクスポートを行い、あるセッションから別のセッションへ、特定のサービスの設定と値を素早く一貫して提供できるようにします。

次の図は、サービステンプレートと、それらに関連付けられている部品を使用するときの一般的なワークフローです。



このガイドでは、Service Builder を使用してサービステンプレートを設計および作成する手順について、サービステンプレート内のステップを作成するための部品も含めて解説します。

Service Builder を使用するには、Ops Center Automator の Develop または Admin ロールが必要です。

権限とユーザーロールに関する詳細情報については、『Hitachi Ops Center Automator ユーザーズガイド』を参照してください。

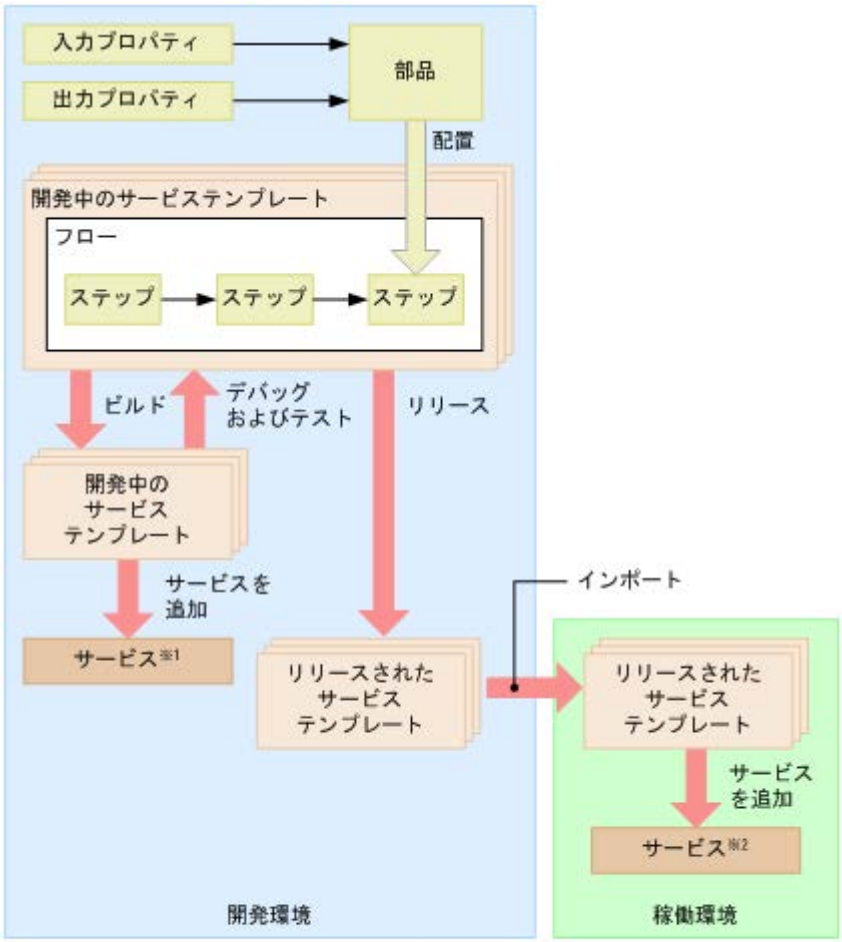
関連概念

- 1.3 Service Builder にアクセスする
- 1.2 用語と概念
- 1.4 インタフェースのナビゲーション

1.2 用語と概念

Service Builder を使い始める前に、各種のコンポーネントを示す用語を学び、それらがどのように関係しているかを理解してください。

次の図は、Service Builder の各種のコンポーネントと、それらの関係を示したものです。



注※1
開発環境でサービステンプレートのテストに使用されるサービス
注※2
稼働環境で実行されるサービス

各用語の定義を以下に示します。

開発環境と稼働環境

サービステンプレートは開発環境で作成され、リリースに適切とみなされるまでテストと修正が行われます。サービステンプレートのデバッグとテストを行った後で、稼働環境へインポー

トします。稼働環境では、ユーザーによりサービスとしてサブミットされた適切なタスクを実行できます。

サービステンプレート

サービステンプレートは、コンポーネント（部品や他のサービステンプレート）により指定される各種の実行可能な運用手順を定義するための枠組みです。Ops Center Automator が提供するパッケージ済みのサービステンプレートは、Service Builder により作成、修正、管理できます。サービステンプレートを作成または修正するときは、開発環境でステップとフローを指定し、関連パラメーターを提供して、十分なデバッグとテストを行います。サービステンプレートが正しく動作し、目的のタスクがエラーなしに実行されるようになった後で、稼働環境へインポートし、サービスとして利用できます。

開発中のサービステンプレート

開発中のサービステンプレートは、開発環境で使われます。運用パラメーターとロジックがコンポーネントのステップで指定されるとき、テスト中、およびテンプレートが最終的にリリースされるまで、そのサービステンプレートは開発中とみなされます。リリース済みのサービステンプレートをコピーして作成されたサービステンプレートも、開発中のサービステンプレートとして分類されます。

リリース済みのサービステンプレート

リリース済みのサービステンプレートは、稼働環境で使われます。関連する部品が定義され、包括的なテストが行われ、稼働環境で使用するためリリースされたサービステンプレートは、リリース済みとみなされます。Service Builder とともに提供されるパッケージ済みのサービステンプレートも、リリース済みのサービステンプレートとして分類されます。リリースされた後のサービステンプレートは直接編集できず、新しいコピーを作成して開発作業を行う必要があります。

コンポーネント

コンポーネントは、フローにステップとして追加できるサービステンプレートまたは部品です。

部品

部品は、サービステンプレートの基本的なコンポーネントです。特定のタスクを実行するためのスクリプトファイル、定義ファイル、リソースファイル、およびアイコンファイルで構成されます。サービステンプレートは複数の部品を含むことができ、互いに結合して一連のタスクを実行します。

入力/出力プロパティ

入力プロパティは、タスクの実行に必要な入力設定と値を指定し、出力プロパティは実行結果を格納して、実行の確認や変更の生成に使用できます。部品の入力プロパティに直接値を入力できます。または、これらをサービスの入力プロパティまたは変数にリンクすることで、これらに値を渡すことができます。サービスの出力プロパティを部品の出力プロパティにリンクすることで、部品の実行結果を確認できます。この方法でプロパティをリンクし、これらの間で値を受け渡しすることを、プロパティマッピングと呼びます。

フロー

フローは処理シーケンスのタスクを定義し、ステップの配置と、それらのステップをリンクするコネクターに従って作成されます。

ステップ

ステップはフローの実行単位で、部品またはサービステンプレートコンポーネントの配置に従って作成されます。コンポーネントは、フローの各ステップで実行されます。ステップは実行時にプロパティ情報を提供します。

サービス

サービスは自動化された命令と、返される出力値のカスタマイズされた組で、コンポーネントのステップのフローと、サービステンプレートでそれらのステップに関連付けられている入力および出力プロパティとにより定義されます。サービスは、Service Builder を使用して作成された、パッケージ済みまたはカスタマイズされたサービステンプレートから生成されます。

タスク

タスクはサービスの実行インスタンスです。タスクは、スケジュールに従って構成され、実行されます。

ビルド/リリース

Service Builder で、すべての処理の定義と構成を完了してから、ビルドを開始すると、必要なすべてのファイルがパッケージ化された、動作するサービステンプレートが作成され、テストやデバッグを行えるようになります。サービステンプレートのテストとデバッグが完了した後で、サービスとしてリリースできます。

インポート/エクスポート

サービステンプレートをリリースするとき、ファイルへエクスポートできます。このファイルを別の Ops Center Automator がインストールされているマシンへインポートして、そのマシン上でサービスとしてサブミットできます。

関連概念

- 1.1 Service Builder について
- 1.3 Service Builder にアクセスする
- 1.4 インタフェースのナビゲーション

1.3 Service Builder にアクセスする

サービスを実行するサービステンプレートの作成、編集、管理は、［Service Builder Home］画面で行います。

Service Builder にアクセスするには、［ツール］プルダウンメニューの［サービスビルダー］をクリックします。

メモ　［ツール］メニューを表示してアクセスするには、Admin または Develop ロールが必要です。

Ops Center Automator へのログイン方法および GUI の他のエリアに関する詳細情報については、『Hitachi Ops Center Automator ユーザーズガイド』を参照してください。

関連概念

- 1.1 Service Builder について

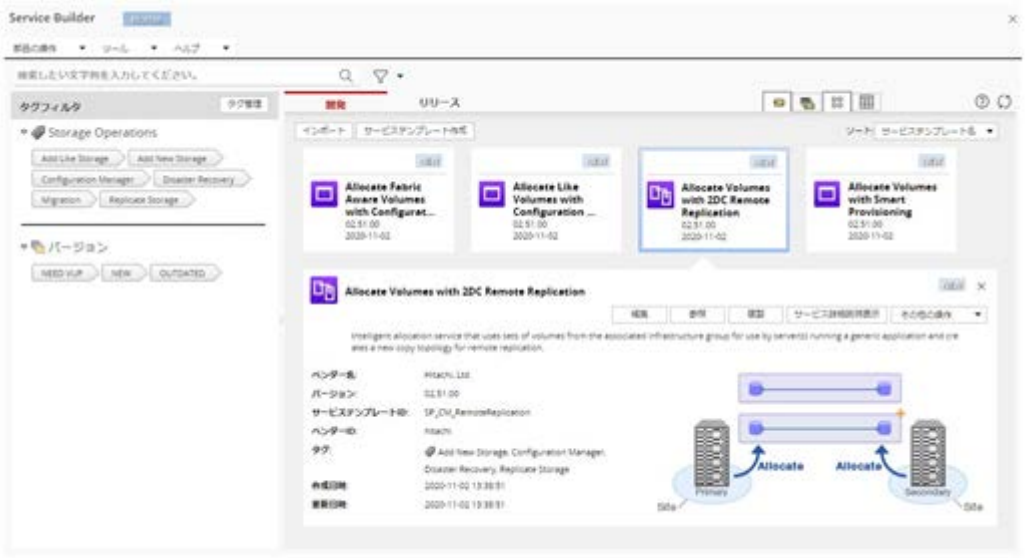
- 1.2 用語と概念
- 1.4 インタフェースのナビゲーション

1.4 インタフェースのナビゲーション

Service Builder のグラフィカルユーザーインタフェース（GUI）は、以下の画面とメニューオプションで構成されています。

［Service Builder Home］画面

［Service Builder Home］画面で、既存のサービステンプレートおよび部品を検索、管理したり、新しいサービステンプレートを作成、編集したりできます。



［Service Builder Home］画面には、すでに作成されているサービステンプレート（［リリース］）または現在開発中のサービステンプレート（［開発］）が表示されます。ここから、選択したサービステンプレートに関連する詳細にすばやくアクセスしたり、管理機能（参照、コピー、編集、削除、インポート、エクスポート）を実行したり、必要な場合は新しいテンプレートを作成したりできます。また、サービステンプレートの機能がビルドされる個々の基本要素としての役割を果たすカスタム部品の操作や作成に使用できるオプションにも［Service Builder Home］画面からアクセスします。

テキスト検索ボックスでは、サービステンプレートを検索できます。また、サービステンプレートが関連付けられているタググルーピングに基づいて検索を実行できます。［カードビュー］または［テーブルビュー］を使用して、サービステンプレートを表示することもできます。

サービステンプレート一覧をソートするには、［ソート:］を選択し、適切なオプションを選択します。

- サービステンプレート名
- ベンダー名
- バージョン
- 説明
- サービステンプレート ID
- ベンダー ID

- 作成日時
- 更新日時
- 適用対象バージョン

サービステンプレートを管理、作成する

多くの場合、既存のサービステンプレートの 1 つに修正を加えて使用することで、使用しているサイトに必要なタスクを実行できます。事前に準備されたサービステンプレートが目的に合わなかった場合は、新しいサービステンプレートを作成できます。

サービステンプレートの管理と作成に次のボタンを使用できます。

- ［インポート］: すでに作成されていながら自動的にインストールされていないサービステンプレート、またはすでに作成されていて別のシステムに保存されているサービステンプレートの 1 つをインポートします。
- ［サービステンプレート作成］: 新しいサービステンプレートを作成し、プロパティ、コマンド、使用しているサイト向けに特別に設計されたスクリプトを含めることができます。
- ［編集］: 選択したサービステンプレートを編集できます。
- ［参照］: 選択したサービステンプレートの詳細を表示します。
- ［複製］: 選択したサービステンプレートをコピーし、編集可能な新しいコピーを作成します。
- ［サービス詳細説明表示］: 選択したサービステンプレートに関連するすべての詳細を示します。
- ［その他の操作］: 次の操作を使用できます。
    - ［エクスポート］: 選択したサービステンプレートの内容を指定したファイルにエクスポートします。
    - ［削除］: 選択したサービステンプレートを削除します。

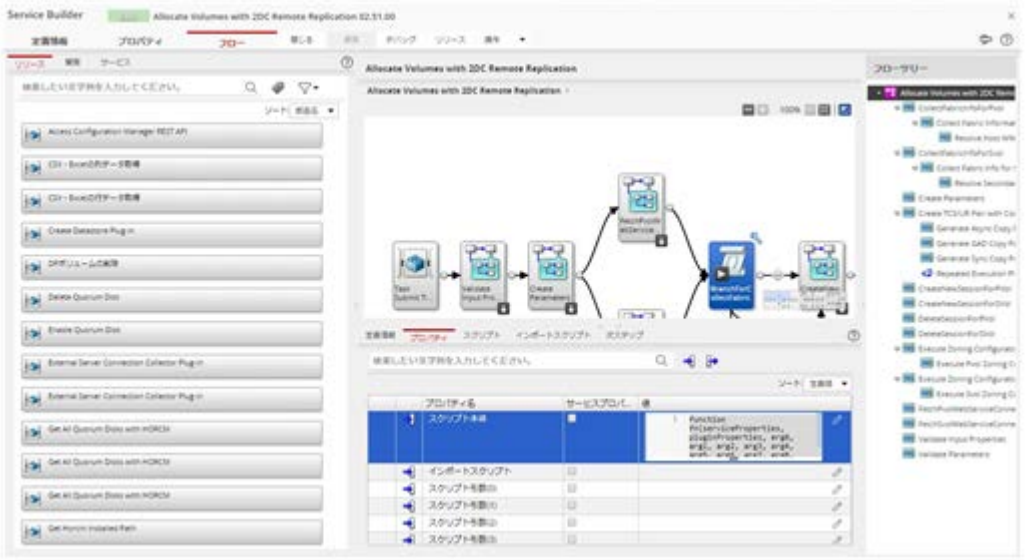
部品を管理、作成する

［部品の操作］プルダウンメニューから適切なオプションを選択して、部品を管理および作成します。

- ［作成］: 新しい部品を作成します。
- ［編集］: 開発状態の既存の部品を変更します。
- ［複製］: リリースまたは開発状態の既存の部品をコピーします。
- ［削除］: リリースまたは開発状態の部品を削除します。

［Service Builder Edit］画面

サービステンプレートの編集は、［Service Builder Edit］画面で行います。この画面では、適切な部品（またはほかのサービステンプレート）を選択し、実行フロー内に配置できます。この場合、与えられたタスクを完了するために必要なロジックと値は、各種の入力プロパティと出力プロパティにより提供されます。



適切なタブをクリックすると、サービステンプレートの詳細およびプロパティ設定を参照および編集できます。

- ［定義情報］タブは、選択したサービステンプレートに関する一般的な詳細を示します。［編集］ボタンをクリックすると、サービステンプレートを編集およびカスタマイズできます。
- ［プロパティ］タブは、選択したサービステンプレートに関連付けられた入力プロパティと出力プロパティを表示します。
  ［追加］プルダウンメニューをクリックすると、サービステンプレートの次のオプションが表示されます。
  - ［プロパティグループ］: 新しいプロパティグループを追加して、各種プロパティを分類します。
  - ［入力プロパティ］: サービステンプレートの新しい入力プロパティを作成および編集します。
  - ［出力プロパティ］: サービステンプレートの新しい出力プロパティを作成および編集します。
  - ［変数］: 新しい変数を作成します。
  - ［サービス共有プロパティ］: よく使用するサービス共有プロパティのコレクションへのアクセスを提供します。

  ［プレビュー］プルダウンメニューから、プレビューを生成できます。このプレビューでは、サービス実行時に、あるプロパティが所定の実行モードに対してどのように処理されるかを（［サービス作成］画面、［サービス実行］画面、または［タスク詳細］画面から）シミュレートできます。
  ［その他の操作］プルダウンメニューでは、以下の追加オプションが提供されます。
  - ［可視性を変更］: サービステンプレートの実行時に、設定を表示するかどうかを制御します。
  - ［表示設定を変更］: 次の表示設定を指定します。

    ［変更可能］: 表示設定を編集できます。

    ［変更不可］: 表示設定を読み取り専用にします。

    ［表示］: 表示設定を表示します。

    ［非表示］: 表示設定を非表示にします。

- ［フロー］タブ：次のパネルが提供されます。
  - 左のパネルには、サービステンプレートのフローに追加できる使用可能なコンポーネント（［リリース］、［開発］、［サービス］）が表示されます。
  - 右上のパネルには、現在選択されているサービステンプレートに関連付けられた部品およびサービステンプレートが表示されます。
  - 右下のパネルには、選択されているコンポーネントのステップで使用できる入力プロパティと出力プロパティ、およびステップの一般的な詳細情報が表示されます。

    ［定義情報］：現在選択されているコンポーネント（部品ステップ）に関する一般的な詳細情報を提供し、使用しているサイト向けにこの情報を編集できます。選択したステップに関連付けられたすべての後続ステップ実行条件も表示されます。

    ［プロパティ］：選択したコンポーネントに関連付けられた入力プロパティと出力プロパティを表示します。これらのプロパティは編集可能です。

    ［次ステップ］：フローの次のステップがどのように実行されるかの条件分岐を作成できます。
  - ［フローツリー］パネルには、現在サービステンプレートに追加されているコンポーネントのステップの構造化されたビューが表示されます。

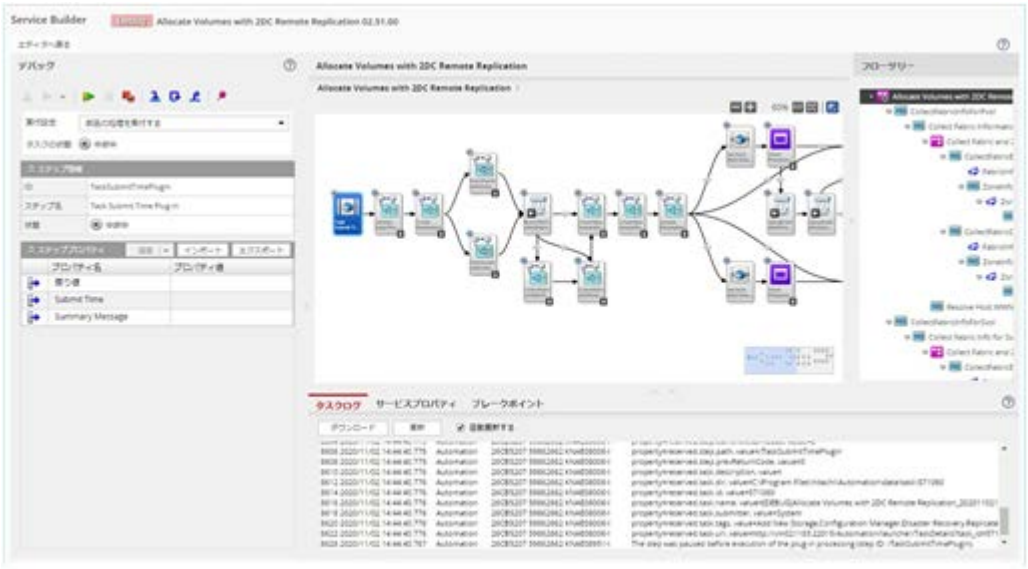
メモ　飛行機のアイコン（［フロー］タブが選択されている場合、［Service Builder Edit］画面の右上）をクリックすると、サービステンプレート作成の一般的なプロセスのツアーを開始できます。また、フローパネルの右上で［縮小図］アイコンをクリックすると、フロー内のステップの概要にアクセスできます。

画面上部にある追加のボタンにより、次の機能が提供されます。

- ［閉じる］：Home 画面に戻ります。
- ［保存］：サービステンプレートを現在の状態で保存します。
- ［デバッグ］：サービステンプレートのビルドと、デバッガーインタフェースへのアクセスを提供します。デバッガーインタフェースでは、タスクを順に実行したり、タスクの実行をシミュレートしたりして、全体が正しく動作することを確認できます。
- ［リリース］：サービステンプレートのデバッグ済みバージョンをリリースします。ユーザーはこのバージョンを、メインの Ops Center Automator ユーザーインタフェースからサービスとしてサブミットできます。
- ［操作］：［コンポーネントバージョン管理］へのアクセスを提供します。ここでは、コンポーネントのバージョンを管理できます。

［Service Builder Debug］画面

［Service Builder Debug］画面から、実行フローを確認し、サービステンプレートをデバッグします。



上部中央のペインには、部品（またはほかのサービステンプレート）が実行されるフローが示され、下部のペインには、次のうちどのタブをクリックしたかに応じて詳細が表示されます。

- ［タスクログ］：実行されたタスクのログを表示します。このログは、ダウンロード（［ダウンロード］）でき、コマンドでリフレッシュ（［更新］）または定期的に自動リフレッシュ（［自動更新する］）できます。
- ［サービスプロパティ］：サービステンプレートに定義された入力プロパティと出力プロパティを表示します。
- ［ブレークポイント］：現在のデバッグセッションで設定されているすべてのブレークポイントを表示します。

右端のペインには、実行されるステップの階層ビューが表示され、左端のペインには、デバッグ制御が提供されます。

アイドルタイムアウトについて

設定したアイドルタイムアウト時間が経過した後に画面を操作すると、エラーメッセージが表示され、ログイン画面に遷移します。したがって、アイドルタイムアウトが発生するように設定している場合は（Common Services の［自動更新］設定が無効、かつ `client.editor.sso.timeout.disable` プロパティの値が［false］になっている）、Service Builder でアイドルタイムアウトが発生する前に、データを適切に保存してください。Service Builder では、`client.editor.sso.timeout.disable` プロパティの値を［true］にすることにより、［自動更新］設定に関わらず、常にアイドルタイムアウトを回避することができます。`client.editor.sso.timeout.disable` プロパティの詳細は、『Hitachi Ops Center Automator インストールガイド』を参照してください。アイドルタイムアウト設定の詳細については、『Hitachi Ops Center Portal ヘルプ』を参照してください。

関連概念

- 1.1 Service Builder について
- 1.3 Service Builder にアクセスする
- 1.2 用語と概念

1.5 開始のためのヒント

初めて実施する複雑な手順の手助けとして、Service Builder はさまざまな手順を視覚的にガイドするオーバーレイを表示します。

例えば、［フロー］タブでサービステンプレートのビルドを最初に開始するときは、便利な［Getting Started Tips］オーバーレイが表示され、部品の検索方法、選択した部品をフローへドラッグアンドドロップする方法、部品に関連付けられているプロパティにアクセスする方法などが提供されます。

ヒントが再度必要になったときは、飛行機のアイコン（［フロー］タブが選択されている場合、［Service Builder Edit］画面の右上）をクリックしてアクティブにできます。

2

既存のサービステンプレートを操作する

［Service Builder Home］画面から適切なオプションを選択して、既存のサービステンプレートを管理できます。

ここでは、サービステンプレートの概要および既存のサービステンプレートを管理する方法について説明します。

2.1 サービステンプレートの概要

カスタムサービステンプレートは、コマンドやスクリプトを実行して特定のタスクセットを自動化する部品（またはほかのサービステンプレート）で構成されています。部品およびサービスをステップとして追加し、順序立てて配置することで操作フローを作成できます。また、サービステンプレートには、データフローを定義する入力および出力プロパティのマッピングが必要です。プロパティグループとサービス共有プロパティを使用すると、入力プロパティと出力プロパティでの定義にさらに役立ちます。

［Service Builder Home］画面にアクセスするときに使用できるサービステンプレートには 2 つのバージョンがあります。

メモ　新しいサービステンプレートを作成すると、デフォルトでサービスの出力プロパティ **service.errorMessage** がサービステンプレートに追加されます。

- ［開発］：新しく作成したサービステンプレートはデバッグバージョンで始まります。サービステンプレートのテストには、サービステンプレートのデバッグバージョンに基づくサービスとタスクの作成が含まれます。サービステンプレートは開発状態となり、コピーして修正することができます。デバッグバージョンのサービステンプレートを再ビルドすると、そのサービステンプレートのこれまでのデバッグバージョンとそのサービスは削除され、関連するタスクはアーカイブされます。開発バージョンのサービステンプレートをリリースすると、そのサービステンプレートのデバッグバージョンとすべての関連するサービスが削除され、タスクがアーカイブされます。
- ［リリース］：リリースバージョンのサービステンプレートはテストが完了しており、リリース状態の下に表示されます。新しいサービスとタスクは、リリース済みのサービステンプレートに基づいて作成および実行できます。リリース済みのサービステンプレートは編集できませんが、別のサービステンプレートで使用するためにコピーして修正することができます。

［Service Builder Home］画面で［開発］または［リリース］タブをクリックして、適切なタイプのサービステンプレートにアクセスできます。［カードビュー］または［テーブルビュー］から特定のサービステンプレートをクリックしてハイライト表示することで、関連項目にアクセスしたり、選択したサービステンプレートの管理機能（参照、編集、コピー、削除、インポート、またはエクスポート）を実行したりできます。必要に応じて、既存のテンプレートの 1 つに基づいて、または一から新しいサービステンプレートを作成し、固有のニーズに合わせて変更することもできます。

関連参照

- 4.1 サービステンプレート作成ワークフロー

2.2 既存のサービステンプレートを管理する

［Service Builder Home］画面から既存のサービステンプレートを管理できます。

既存のサービステンプレートを管理する手順は、このセクションで説明しています。

2.2.1 サービステンプレートを表示する

開発またはリリース状態のサービステンプレートは、［Service Builder Home］画面で参照できます。

既存のサービステンプレートを表示するには、次の手順に従います。

操作手順

1. ［Service Builder Home］画面で、［テーブルビュー］または［カードビュー］から目的のサービステンプレートを選択してハイライト表示し、使用可能な操作を表示します。
   選択したサービステンプレートの詳細と使用できるオプションが表示されます。
2. ［参照］をクリックします。

操作結果

［Service Builder View］画面が、選択したサービステンプレートのフローを表示する［フロー］タブが選択された状態で表示されます。［定義情報］タブおよび［プロパティ］タブをクリックして、選択したサービステンプレートおよびそれに関連付けられたプロパティに関する詳細を参照できます。

2.2.2 サービステンプレートをコピーする

サービステンプレートをコピー、編集すると、既存のサービステンプレートの新バージョンを作成でき、これを特定の運用環境用に修正できます。

既存のサービステンプレートをコピーするには、次の手順に従います。

操作手順

1. ［Service Builder Home］画面で、［テーブルビュー］または［カードビュー］から目的のサービステンプレートを選択してハイライト表示し、使用可能な操作を表示します。
   選択したサービステンプレートの詳細と使用できるオプションが表示されます。
2. ［複製］をクリックします。
   ［サービステンプレート複製］ダイアログが表示されます。
3. ［サービステンプレート複製］ダイアログで、サービステンプレートの基本情報を入力し、［OK］をクリックします。
   選択したサービステンプレートのコピーが作成されます。

操作結果

［Service Builder Edit］画面が、新規作成したサービステンプレートのフローを表示する［フロー］タブが選択された状態で表示されます。［定義情報］タブおよび［プロパティ］タブをクリックして、選択したサービステンプレートおよびそれに関連付けられたプロパティに関する詳細を参照できます。

関連参照

- 2.2.3 ［サービステンプレート複製］ダイアログ

2.2.3 ［サービステンプレート複製］ダイアログ

［サービステンプレート複製］ダイアログには、新たにコピーされたサービステンプレートの詳細が表示されます。

次の表で、［サービステンプレート複製］ダイアログのフィールド、サブフィールド、およびフィールドグループについて説明します。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

表 1 ［サービステンプレート複製］ダイアログ

フィールド	サブフィールド	説明
［サービステンプレート ID*］	-	コピーしたサービステンプレートのキー名を指定します。

フィールド	サブフィールド	説明
［サービステンプレートバージョン*］	-	サービステンプレートのバージョン。
［ベンダー ID*］	-	ベンダー ID。
［サービステンプレート名*］	-	ユーザーインタフェースを通じて示されるサービステンプレートの名前。
［ベンダー名］	-	もし分かれば、コピーして作成したサービステンプレートのベンダー名。
［説明］	-	コピーしたサービステンプレートの説明。
［タグ］	-	サービステンプレートに関連付けられたタグカテゴリ。

アスタリスク（*）は必須フィールドであることを示します。

関連タスク

- 2.2.2 サービステンプレートをコピーする

2.2.4 サービステンプレートを編集する

既存のサービステンプレート（コピーしたテンプレートで開発状態のもの）を編集して、詳細を修正、追加できます。

前提条件

- サービステンプレートの目的を達成するには、どのコンポーネントが必要かを明らかにします。
- 必要なコンポーネント（リリースまたは開発状態の部品やほかのサービステンプレート）をサービステンプレートで使用できるようにします。

操作手順

1. ［Service Builder Home］画面の［開発］タブで、編集するテンプレートを選択してハイライト表示します。
2. ［編集］ボタンをクリックします。
   ［Service Builder Edit］画面が、編集するテンプレートに関連付けられたコンポーネント（リリースまたは開発状態の部品およびほかのサービステンプレート）を示す［フロー］タブが選択された状態で表示されます。［定義情報］タブまたは［プロパティ］タブをクリックして、選択したサービステンプレートに関する詳細をさらに編集できます。

次の作業

コンポーネントを追加または変更し、入力と出力のステッププロパティを指定し、フローを作成することで、サービステンプレートの編集を続けます。

関連参照

- 4.3 ［サービステンプレート作成］ダイアログ

2.2.5 サービステンプレートを削除する

開発状態のサービステンプレートを削除できます。

リリース状態のサービステンプレートを削除するには、`deleteservicetemplate` コマンドを実行します。Ops Center Automator でコマンドを実行する方法については、『Hitachi Ops Center Automator ユーザーズガイド』を参照してください。

前提条件
開発状態のサービステンプレートを削除する前に、以下の手順を完了します。

1. サービステンプレートに関連するすべての実行中のタスクを停止します。
2. サービステンプレートに関連するすべてのタスクをアーカイブします。
3. サービステンプレートに関連するすべてのサービスを削除します。

メモ　サービステンプレートを削除する前にサービステンプレートに関連するすべてのタスクを停止してアーカイブすることと、サービステンプレートに関連するすべてのサービスを削除することが重要です。削除したサービステンプレートを復元することはできません。

この手順を実行する方法については、『Hitachi Ops Center Automator ユーザーズガイド』を参照してください。

操作手順

1. ［Service Builder Home］画面で、［テーブルビュー］または［カードビュー］から目的のサービステンプレートを選択してハイライト表示し、使用可能な操作を表示します。
   選択したサービステンプレートの詳細と使用できるオプションが表示されます。
2. ［その他の操作］プルダウンから、［削除］オプションを選択します。
   確認を求める［削除］メニューが表示されます。
3. ［OK］をクリックして削除を確認します。
   指定したテンプレートが削除されたかどうかを示す［情報］メニューが表示されます。
4. ［OK］をクリックします。

操作結果

指定したサービステンプレートが削除されます。

2.2.6 サービステンプレートをインポートする

サービステンプレートをインポートし、そのサービスをシステムで使用できます。

前提条件

インポートするサービステンプレートにローカルシステムまたはネットワーク経由でアクセスできる必要があります。サービステンプレートを他のシステムからエクスポートすることで、そのサービステンプレートをインポートできるようになります。

メモ　Ops Center Automator にデフォルトで付属するサービステンプレートに加えて、インポート可能な他のサービステンプレートのコレクションがあります。これらのテンプレートはリリース済みの状態であり、すぐに使用できます。ビルドプロセスを経由する必要はありません。

サービステンプレートをインポートするには、次の手順に従います。

操作手順

1. ［Service Builder Home］画面の［開発］タブで、［インポート］ボタンをクリックします。
   ［サービステンプレートインポート］ダイアログが表示されます。
2. ［選択］をクリックし、インポートするテンプレートの名前と場所を指定します。
3. テンプレートの名前と場所を指定してから、［OK］をクリックします。

操作結果

指定したサービステンプレートがインポートされます。

メモ　サービステンプレートパッケージをインポートすると、サービステンプレート内のサービスコンポーネントがサービステンプレートとしてインポートされます。

関連参照

- 2.2.7 ［サービステンプレートインポート］ダイアログ
- 付録 A.1 ビルトインサービステンプレートの一覧

2.2.7 ［サービステンプレートインポート］ダイアログ

サービステンプレートを 1 つのシステムからエクスポートして現在のシステムにインポートできます。

次の表で、［サービステンプレートインポート］ダイアログのフィールド、サブフィールド、およびフィールドグループについて説明します。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

表 2　［サービステンプレートインポート］ダイアログ

フィールド	サブフィールド	説明
［ファイル］	-	［選択］ボタンをクリックし、サービステンプレートのインポート先の場所とファイル名を指定します。

関連タスク

- 2.2.6 サービステンプレートをインポートする

2.2.8 サービステンプレートをエクスポートする

サービステンプレートをエクスポートすると、別のシステムでそのサービスを使用できます。

前提条件

エクスポートするサービステンプレートにローカルシステムまたはネットワーク経由でアクセスできる必要があります。

操作手順

1. ［Service Builder Home］画面の［開発］タブで、［テーブルビュー］または［カードビュー］から目的のサービステンプレートを選択してハイライト表示し、使用可能な操作を表示します。
   選択したサービステンプレートの詳細と使用できるオプションが表示されます。
2. ［その他の操作］をクリックして、プルダウンメニューから［エクスポート］オプションを選択します。
   ［エクスポート］メニューが表示されます。
3. サービステンプレートのエクスポート先の名前と場所を入力し、［OK］をクリックします。

操作結果

選択したサービステンプレートがエクスポートされます。

3

既存の部品を操作する

部品は、サービステンプレートを作成するための基本的な構成要素です。それぞれの部品は特定の目的のために設計されており、サービステンプレートでの部品の使用および順序によって、特定のタスクの実行順序が決定されます。既存の部品を使用したり、ニーズに合わせて編集したりできます。

ここでは、部品の概要および部品を管理する方法について説明します。

3.1 部品の概要

1 つまたは複数の部品（またはその他のリリース済みサービステンプレート）をサービステンプレートに挿入すると、コマンドまたはスクリプトを実行できます。入力および出力プロパティとリモートコマンドを部品内に設定できます。部品のリモートコマンドを使用する場合、コマンドまたはスクリプトの引数として入力プロパティを指定することで、入力プロパティをコマンドまたはスクリプトに渡すことができます。部品をサービステンプレートの中で編成することで、フローを作成できます。部品は、割り当て先のサービステンプレートと共にテストしてリリースします。部品に加えて、サービステンプレートのフローにほかのサービステンプレートを追加することもできます。

部品のタイプ

部品には、次のように 2 つのタイプがあります。

- ［リリース］: リリース済み部品には、サービステンプレート内でリリースされているカスタム部品が含まれます。開発中のサービステンプレートがリリースされると、そのサービステンプレートに含まれている部品はリリース済みの部品になります。リリース済みの部品は編集できませんが、他のサービステンプレートで使用するためにコピーして修正することができます。［Service Builder Edit］画面の［フロー］タブから使用できる［リリース］タブに、リリース済みの部品が一覧表示されます。
- ［開発］: 新規作成された部品とビルドプロセスおよびテストが完了していない部品は、開発状態の部品です。開発状態の部品は、コピーして変更し、ほかのサービステンプレートで使用できます。作成またはテスト段階では、開発状態の部品は、［Service Builder Edit］画面の［フロー］タブから使用できる［開発］タブに表示されます。

部品に加えて、別のサービステンプレートのフローにサービステンプレートをコンポーネントとして追加することもできます。

- ［サービス］: サービスコンポーネントは、Ops Center Automator にインポートされたリリース済みのサービステンプレートです。サービスをコンポーネントとして使用し、別のサービステンプレートのフローに配置すると、新しいサービステンプレートにサービスコンポーネントのフローを組み込むことができます。Ops Center Automator では、ビルトインのサービスコンポーネントとそのビルトインのサービスのセットが提供されています。［Service Builder Edit］画面の［フロー］タブから使用できる［サービス］タブに、サービスコンポーネントが表示されます。

関連参照

- 3.2 ［部品一覧］ダイアログ
- 5.1 部品作成のワークフロー

3.2 ［部品一覧］ダイアログ

［部品一覧］ダイアログには、［部品の操作］プルダウンメニューから選択したオプション（複製、編集、または削除）に応じた部品が表示されます。識別テキストを入力するか、部品が関連付けられているタグに従って、既存の部品を検索できます。［リリース］タブで既存の部品を管理したり、［開発］タブで現在開発中の部品にアクセスしたりできます。［カードビュー］または［テーブルビュー］ボタンをクリックして、画面に部品を一覧表示する方法を指定することもできます。

次の表で、ダイアログのフィールド、サブフィールド、およびフィールドグループについて説明します。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

表 3 ［部品一覧］ダイアログ

フィールド	サブフィールド	説明
［ベンダー名］	-	もし分かればベンダー名。
［バージョン］	-	部品のバージョン。
［コンポーネント ID］	-	部品と関連するキー。
［ベンダー ID］	-	部品と関連するベンダー ID。
［タグ］	-	部品と関連するタグカテゴリ。
［作成日時］	-	部品が登録された日時。
［更新日時］	-	部品が最後に更新された日時。

関連概念

- 3.1 部品の概要
- 4.1 サービステンプレート作成ワークフロー

3.3 既存の部品を管理する

［部品の操作］プルダウンメニューから適切なオプションを選択することにより、Service Builder の［部品一覧］ダイアログで既存の部品を管理できます。

3.3.1 部品をコピーする

部品のコピーを作成してから、運用サイトに固有の要件に合わせて必要な修正を行うことができます。

開発またはリリース状態にあるどの部品でもコピーできます。既存の部品をコピーするときは、新しいベンダー ID、部品名、バージョン番号を割り当てる必要があります。

操作手順

1. ［Service Builder Home］画面で、［部品の操作］プルダウンメニューから［複製］オプションを選択します。
   ［部品一覧］ダイアログが表示されます。
2. ［部品一覧］ダイアログで、［リリース］または［開発］タブから、コピーする部品を選択してハイライト表示し、［複製］をクリックします。
   ［部品複製］ダイアログが表示されます。
3. ［部品複製］ダイアログで、［定義情報］、［プロパティ］、または［リモートコマンド］タブから、該当の部品情報を入力し、［保存］をクリックします。
   部品のコピーが作成され、［部品一覧］ダイアログの［開発］タブから使用できるようになります。

次の作業

必要に応じて、［部品の操作］プルダウンメニューから［編集］オプションを選択することにより、コピーした部品の操作を続けることができます。

関連参照

- 3.3.3 ［部品複製］ダイアログ

3.3.2 部品を編集する

開発状態の部品に関連付けられている入力プロパティと出力プロパティ、変数、リモートコマンドを編集できます。

操作手順

1. ［Service Builder Home］画面の［部品の操作］プルダウンメニューから、［編集］オプションを選択します。
   ［部品一覧］ダイアログが表示され、編集する部品を選択できます。
2. ［部品一覧］ダイアログで、編集するカスタム部品を選択し、［編集］をクリックします。
   選択した部品の［部品編集］ダイアログが表示されます。
3. 適切なタブ（[定義情報]、［プロパティ］、または［リモートコマンド]）から設定を選択して、選択した部品を編集します。編集が完了したら、［保存］をクリックします。
   編集した部品が保存されたことを示す情報メッセージが表示されます。

3.3.3 ［部品複製］ダイアログ

［部品複製］ダイアログは、部品のコピーを作成するとき、詳細の指定に使用します。

次の表では、現在選択されているタブに基づいて、使用できるダイアログのフィールド、サブフィールド、およびフィールドグループを説明します。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

［定義情報］タブで、選択した部品に関する詳細を参照、入力できます。

表 4 ［定義情報］タブ

フィールド	サブフィールド	説明
［部品 ID:］*	-	コピーして作成した部品のキー名を指定します。
［部品バージョン:］*	-	部品のバージョン。
［ ベンダー ID:］*	-	部品に割り当てられたベンダー ID。
［部品名:］*	-	ユーザーインタフェースを通じて示される部品名。
［ベンダー名:］	-	もしわかれば、コピーして作成した部品のベンダー名。
［説明:］	-	コピーして作成した部品の説明。
［タグ:］	-	部品と関連するタグカテゴリ。
［アイコン:］	-	部品と関連するアイコン。

アスタリスク（*）が付いたフィールドは必須です。

部品のコピーを作成する際に、オリジナルとコピーしたものとを区別するために部品 ID またはベンダー ID を変更する必要があります。

［プロパティ］タブで、選択した部品に関連付けられたすべての入力プロパティと出力プロパティを検索、参照できます。適切なオプションをクリックすることにより、プロパティを管理することもできます。例えば、［追加］ボタンを選択して新しい入力プロパティまたは出力プロパティを追加したり、［編集］ボタンを選択して既存のプロパティを編集したり、［削除］ボタンを選択して選択したプロパティを削除したりできます。

表 5 ［プロパティ］タブ

フィールド	サブフィールド	説明
［プロパティキー］	-	入力プロパティまたは出力プロパティのキー名を指定します。
［プロパティ名］	-	入力プロパティまたは出力プロパティの表示名を指定します。
［説明］	-	入力プロパティまたは出力プロパティの説明とその機能を提供します。
［必須区分］	-	入力プロパティまたは出力プロパティが必須（true）か必須ではない（false）かを示します。
［デフォルト値］	-	入力プロパティまたは出力プロパティに関連付けられたデフォルト値を示します。

［リモートコマンド］タブで、［プラットフォームを追加］プルダウンメニューから適切なオプションを選択することにより、リモートコマンドを実行するプラットフォームを選択できます。［エージェントレス接続先設定を使用］または［プロパティで指定］を選択することにより、［認証種別］も指定できます。選択したプラットフォームに応じて、リモートコマンドの実行方法を制御するほかのオプションが提供されます。

表 6 ［リモートコマンド］タブ

フィールド	サブフィールド	説明
［認証種別］	-	部品で必要とされる認証情報タイプを指定します。 サービスを実行するとき、［管理］タブの［エージェントレス接続先定義］ビューにある認証情報を使用する場合は、［エージェントレス接続先設定を使用］を選択します。認証情報タイプのデフォルト値は［エージェントレス接続先設定を使用］です。 ［エージェントレス接続先設定を使用］の認証情報タイプでは、次の予約済みの部品プロパティが自動的に設定されます。 plugin.destinationHost IPv4 アドレス、IPv6 アドレス、またはホスト名（最大 256 文字）を使用して操作の対象を入力します。 ［プロパティで指定］を選択して、認証情報を入力プロパティとして使用します。 次の予約部品プロパティは［プロパティで指定］の入力プロパティの認証情報タイプに自動的に設定されています。 ・ plugin.destinationHost IPv4 アドレス、IPv6 アドレス、またはホスト名（最大 256 文字）を使用して操作の対象を入力します。宛先のホストが Ops Center Automator サーバ（localhost）ならば、ユーザー ID とパスワードは必要ありません。 ・ plugin.account 対象ホストにログインするためのユーザー ID を入力します（最大 256 文字）。 ・ plugin.password 対象ホストにログインするためのパスワードを入力します（最大 256 文字）。 ・ plugin.suPassword Linux 環境で対象ホストにログインするために使用する root アカウントのパスワードを入力します（最大 245 文字）。対象

フィールド	サブフィールド	説明
		ホストで Windows が実行されている場合、このプロパティは無視されます。
[Windows 接続用設定]	[システムアカウントで実行]	システムアカウントを使用してコマンドを実行します。
[Linux/UNIX 接続用設定]	[root 権限で実行]	root 権限を使用して実行します。
	[文字セット自動判定]	文字セット自動判定は Linux オペレーティングシステムに適用されます。有効にすると、ユーザーのデフォルトのロケールを使用してスクリプトが実行されます。無効にすると、LC_ALL=C のロケールでスクリプトが実行されます。デフォルト値は「有効」です。
[プラットフォームを追加]	-	リモートコマンドのプラットフォーム（Windows または Linux）を指定します。
[定義情報の取り込み]	-	別のオペレーティングシステムから取得する必要のある設定を指定します。
[実行モード:]	-	スクリプトに基づく場合は［スクリプト］を、保存したコマンドを使用する場合は［コマンド］を選択します。
[コマンドライン:] *	-	リモートコマンドのコマンド (8,192 文字まで) を入力します。［実行モード］で［スクリプト］または［コマンド］を選択した場合は必須です。［入力プロパティを挿入］ボタンは、その部品について定義されているすべての入力プロパティをコマンドに挿入します。
[スクリプト設定:]	-	スクリプトファイルをアップロードする場合は［添付］を選択し、スクリプト情報を直接入力するには［直接入力］を選択します。スクリプトを実行する部品では、次のファイル形式を使用します。*<部品名>.<拡張子>*.
[ファイル] *	-	［選択］ボタンをクリックして、コマンドまたはスクリプトが格納されているファイルを指定します。
[出力プロパティのマッピング定義一覧]	-	コマンドまたはスクリプトの出力プロパティのマッピング定義を指定します。一覧からプロパティを選択してハイライト表示し、［鉛筆］アイコンをクリックすることで、［出力フィルタ編集］ダイアログにアクセスでき、出力プロパティに渡すデータを制御する出力フィルタを指定できます。
[実行ディレクトリ]	-	実行フォルダを指定します。
[環境変数]	-	コマンドまたはスクリプトの環境変数を選択します。このオプションを選択すると［環境変数作成］または［環境変数編集］ダイアログが表示され、適切な変数情報を入力できます。

アスタリスク（*）が付いたフィールドは必須です。

関連タスク

- 3.3.1 部品をコピーする

3.3.4 部品を削除する

不要になった部品を削除できます。削除した部品を復元することはできません。

操作手順

1. ［Service Builder Home］画面の［リリース］または［開発］タブで、［部品の操作］プルダウンメニューから［削除］オプションを選択します。

［部品一覧］ダイアログが表示されます。

2. ［カードビュー］または［テーブルビュー］で、削除する部品をクリックしハイライト表示して、［削除］をクリックします。
   意思を確認する［削除］ダイアログが表示されます。
3. ［OK］をクリックして、選択した部品の削除を確認します。

操作結果

部品が削除され、［Service Builder Home］画面に表示されなくなります。

関連タスク

- 3.3.1 部品をコピーする

4

新しいサービステンプレートを作成する

カスタムサービステンプレートを作成するには、まず新しいテンプレートを作成するか、または既存のテンプレートをコピー・編集します。そのあと、新規作成または修正したサービステンプレートは、コンポーネント（部品やほかのサービステンプレート）、リソースファイル、アイコンファイル、カスタムファイル、フロー、およびサービス定義により定義されます。

ここでは、サービステンプレートを作成し編集する方法について説明します。

4.1 サービステンプレート作成ワークフロー

［Service Builder Home］画面から新しいサービステンプレートを管理、作成できます。（［部品の操作］プルダウンメニューから、新しい部品を管理、作成することもできます。）

既存のサービステンプレート（編集、参照、コピー、インポート、エクスポートなど）を管理するには、サービステンプレートを（［開発］または［リリース］タブから）選択し、実行する操作のタイプに適したボタンをクリックします。サービステンプレートに関する詳細を確認するには、［サービス詳細説明表示］ボタンをクリックします。

新しいサービステンプレートを作成する場合、ある程度計画を立ててから、次のワークフローで説明しているさまざまなフェーズを経る必要があります。

フェーズ 1：準備

1. サービステンプレートの必要性と目的を決定します。プロセスを自動化するのに必要なステップを検討し、ステップに新しいテンプレートの作成と既存テンプレートの修正のどちらが必要かを決定します。
2. サービステンプレートの作成の準備をします。これには既存のコンポーネントの特定、もしくは新規コンポーネントの作成、アイコンファイルの準備、および定義、リソースファイル、カスタムファイルおよびフローの設定が含まれます。

フェーズ 2：作成

1. 新しいサービステンプレートを作成するか、要件に合えば既存のサービステンプレートをコピーして修正します。名前、ID、ベンダー名、および説明などの基本情報を入力します。この段階では、サービステンプレートは開発状態になります。
2. 新しい部品を作成するか、目的を満たすものがあれば既存の部品を使用します。
3. テンプレートと部品をステップとして配置し、実行される順序で接続することでサービステンプレートのフローを定義します。
4. コンポーネントの入力および出力プロパティをサービステンプレートの入力および出力プロパティにマッピングすることで、データフローを定義します。
5. サービステンプレートの入力および出力プロパティを使用してサービス定義を設定します。サービス共有プロパティ、プロパティグループ情報、および変数を追加します。

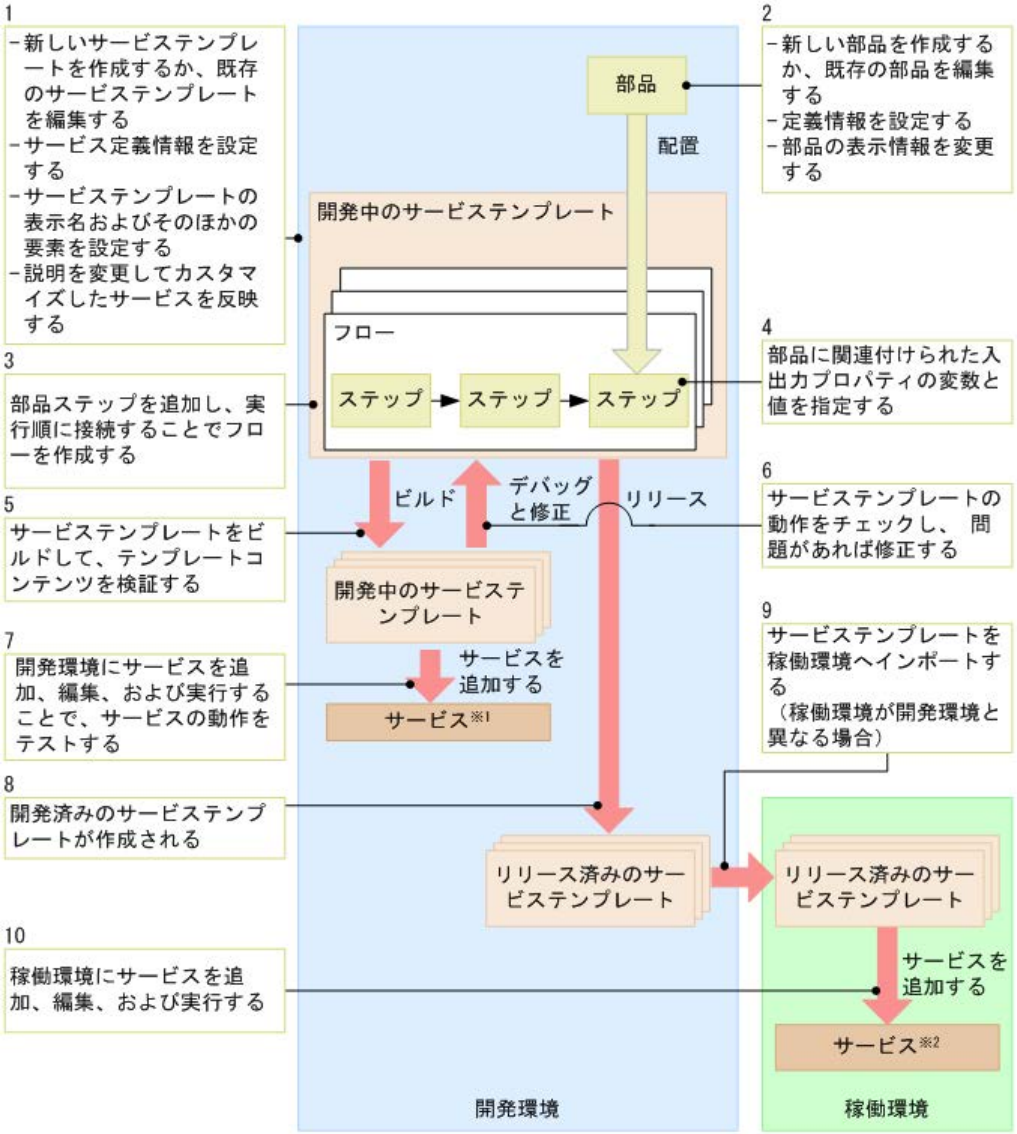
フェーズ 3：テストおよびデバッグ

1. サービステンプレートをテスト用にビルドします。
2. テストを実行します。サービステンプレートのデバッグ構成に基づいてサービスを作成します。
3. テスト結果として修正を行います。
4. 正しく機能するようになるまでサービステンプレートの再ビルドと再テストを行います。

フェーズ 4：リリース

サービステンプレートをリリースします。操作環境にサービステンプレートをサブミットするには、サービステンプレートがリリース状態である必要があります。

次の図に、サービステンプレートを作成または編集するときに従う一般的な手順を示します。



注※1
開発環境のテンプレートをテストするサービス
注※2
稼働環境で実行されたサービス

関連参照

- 3.2 ［部品一覧］ダイアログ
- 6.1 デバッグとリリースのワークフロー

4.2 新しいサービステンプレートを作成する

新しいサービステンプレートを作成して、特定のタイプのユーザー向けに一連のタスクを自動化するようにカスタマイズしたサービスを作成できます。

前提条件

- サービステンプレートの目的を達成するには、どのコンポーネントが必要かを明らかにします。
- コンポーネント（リリースまたは開発状態）をサービステンプレートで利用可能にします。

メモ　ここでは新しいサービステンプレートを作成する手順を説明していますが、通常はモデルとして使う既存のサービステンプレートの 1 つを単にコピーして、次に特定の運用環境に合わせて必要な修正を行うことができます。

操作手順

1. 次の方法から 1 つを選び、新しいサービステンプレートを作成します。
   - 一から新しいサービステンプレートを作成するには、[Service Builder Home] 画面の [開発] タブで、[サービステンプレート作成] ボタンをクリックします。
   - 既存のテンプレートに基づいて新しいサービステンプレートを作成するには、[Service Builder Home] 画面の [開発] タブまたは [リリース] タブで、[複製] ボタンをクリックします。

   一から新しいサービステンプレートを作成している場合、[サービステンプレート作成] ダイアログが表示されます。サービステンプレートが既存のテンプレートに基づく場合、[サービステンプレート複製] ダイアログが表示されます。
2. 新しいサービステンプレートの情報を入力し（テンプレートを関連付けるタグを選択し）、[OK] をクリックします。
   [Service Builder Edit] 画面が、[フロー] タブが選択された状態で表示されます。

操作結果

ここから、必要なコンポーネント（リリースまたは開発状態の部品およびほかのサービステンプレート）の検索と、コンポーネントの実行順序に従ったコンポーネントのフローへのドラッグを開始できます。

次の作業

入力と出力のステッププロパティの指定とフローの作成を続けて、終了すると、サービステンプレートをデバッグ、リリースできます。

関連参照

- 4.3 [サービステンプレート作成] ダイアログ

4.3 [サービステンプレート作成] ダイアログ

新しいサービステンプレートを作成するときは、[サービステンプレート作成] ダイアログで適切な詳細情報を入力します。

次の表で、ダイアログのフィールド、サブフィールド、およびフィールドグループについて説明します。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 7 [サービステンプレート作成] ダイアログ

フィールド	サブフィールド	説明
[サービステンプレート ID*]	-	アクセスと追跡を容易にするために各サービステンプレートに一意のキー名を割り当てます。
[サービステンプレートバージョン*]	-	新しいサービステンプレート用のバージョン番号。

フィールド	サブフィールド	説明
[ベンダー ID *]	-	ベンダーに関連付けられた ID。
[サービステンプレート名: *]	-	ユーザーインタフェースを通じて示される新しいサービステンプレートの名前。
[ベンダー名]	-	もし分かればベンダー名。
[説明 :]	-	新しいサービステンプレートの説明。
[タグ :]	-	テンプレートが関連付けられたタグカテゴリを指定します。

アスタリスク（*）が付いたフィールドは必須です。

関連タスク

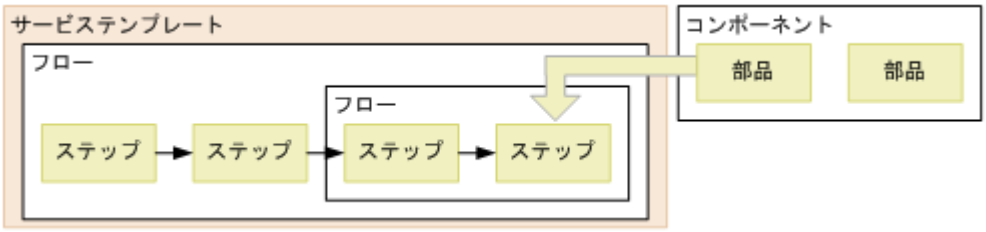
- 4.2 新しいサービステンプレートを作成する
- 2.2.4 サービステンプレートを編集する

4.4 ステップフローを指定する

コンポーネント（部品またはほかのサービステンプレート）を追加し、[フロー] タブでコンポーネントの実行順序を決定することで、サービステンプレートのステップを指定します。

サービステンプレートのステップのフローを作成する場合は、[フロー] ビューから適切な部品（またはサービステンプレート）のステップを追加し、コネクターの線でつなげてステップの実行順序を設定します。次に入力および出力プロパティのマッピングを指定する必要があります。

次の図に、サービステンプレートの一般的なフローを示します。



[フロー] タブでサービステンプレートのビルドを最初に開始するときは、便利な「ツアー」オプションが表示され、部品の検索手順、選択した部品をフローへドラッグアンドドロップする方法、部品に関連付けられているプロパティにアクセスする方法を説明します。このツアーを最初に確認したら、適切なチェックボックスをオンにして、ツアーがその後表示されないよう指定することができます。必要なときはいつでも、ツアーを再度アクティブにできます。

4.4.1 データフローでステップを作成する

サービステンプレートにステップを作成するには、[Service Builder Edit] 画面の [フロー] タブで、サービスの適切なコンポーネントを指定します。

前提条件

- 編集または作成を行うサービステンプレートは開いており、開発状態にある必要があります。
- サービステンプレートの目的を達成するには、どの部品が必要かを明らかにします。
- コンポーネント（リリースまたは開発状態）をサービステンプレートで利用可能にします。

操作手順

1. ［Service Builder Edit］画面の［フロー］タブで、コンポーネントをクリックし［フロー］ビューへドラッグして、ボタンを離します。
   ［ステップ作成］ダイアログが表示されます。
2. ステップの情報を入力し、［OK］をクリックします。
   ［フロー］ビューに、コンポーネントのアイコンがステップとして表示され、［定義情報］または［プロパティ］タブから、ステップに関する詳細にその関連プロパティも含めてアクセスできます。
3. 必要に応じてステップの追加を続けます。
   - ステップを編集、コピー、切り取り、または削除するには、コンポーネントを右クリックして適切なオプションを選択します。コンポーネントをコピーする場合、別のサービステンプレートにペーストすることができます。
   - 複数のステップを選択するには、マウスをドラッグして対象のステップ群を取り囲む四角形を作成するか、または各ステップアイコンを［Ctrl］キーを押しながらクリックして選択を追加します。
   - ステップを移動するには、そのステップのアイコンをクリックし、［フロー］ビューの目的のエリアへドラッグして、ボタンを離します。
   - 2 つ目のフローを追加するには、［フローツリー］ビューで対象のフローを右クリックし、［階層作成］をクリックします。
4. 変更を保存しないで［Service Builder Edit］画面を終了するには、［閉じる］ボタンをクリックします。変更を保存するには、［保存］をクリックします。

操作結果

ステップがサービステンプレートに追加されて保存されました。

次の作業

続けてサービステンプレートを編集し、ステップが実行される順序を示すフローを追加します。

関連参照

- 4.4.2 ［ステップ作成］または［ステップ編集］ダイアログ

4.4.2 ［ステップ作成］または［ステップ編集］ダイアログ

サービステンプレートのステップに対して、コンポーネントに関連付けられた詳細を［ステップ作成］または［ステップ編集］ダイアログで入力できます。

メモ　［参照］ボタンをクリックして、ステップのコンポーネントに関する詳細を取得できます。

次の表は、［ステップ作成］または［ステップ編集］ダイアログのフィールド、サブフィールド、フィールドグループの説明です。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 8 ［ステップ作成］または［ステップ編集］ダイアログ

フィールド	サブフィールド	説明
［ステップ］	［ステップ ID］*	ステップの ID。
	［ステップ名］*	ステップの名前。
	［説明］	ステップの短い説明。
［コンポーネント］	［ベンダー名］	コンポーネントと関連するベンダー名。
	［部品バージョン］	部品のバージョン。ドロップダウンボックスをクリックすると、複数のバージョンがあれば選択項目を表示します。
	［部品 ID］	コンポーネントと関連したキー名。
	［ベンダー ID］	部品のベンダー名。
	［タグ］	ステップに関連付けられたタグカテゴリ。
	［作成日時］	ステップが最初に登録された日時。
	［更新日時］	ステップが最後に更新された日時。
［ 後続ステップ実行条件］	［条件：］	いつ次のステップを実行するかの指示を設定します。選択肢は次のとおりです。 • ［戻り値をしきい値により判定する］ 戻り値が［判定値］と同じか、それより少ない時に次のステップを実行します。戻り値が［判定値］以上であれば、このステップはエラー状態で終了します。 • ［戻り値によらず常に正常終了］ 常に次のステップを実行します。 • ［戻り値によらず常に異常終了］ このステップの最後で常にエラー状態で終了します。
	［判定値：］*	一定の条件ステップが実行されるしきい値となる整数を 0～255 の中から設定します。
	［警告設定：］	この設定が有効な場合、［警告設定］を超えた場合には次のステップがエラー状態で実行されます。
	［警告値：］	［戻り値をしきい値により判定する］が有効な場合に指定する必要があります。整数を 0～255 の中から設定します。戻り値が［警告値］および［判定値］よりも小さい場合、次のステップを実行します。戻り値が［警告値］以上で、［判定値］以下の場合は、タスクの状態は実行中は「異常検出」を、タスクの完了後は「失敗」を示します。［警告値］を［判定値］よりも大きく設定することはできません。

アスタリスク（*）は必須フィールドであることを示します。

関連タスク

- 4.4.1 データフローでステップを作成する

4.4.3 ステッププロパティを指定する

コンポーネント（部品またはほかのサービステンプレート）に関連付けられたタスクを実行する各種の入力プロパティと出力プロパティを指定する必要があります。

サービステンプレートは汎用的な操作プロシージャを定義します。このため、ホスト名やリソース制限などサービスを実行するのに必要な入力値を格納するプロパティは、サービステンプレートからサービスが追加されたときに定義されます。これらはサービス入力プロパティと呼ばれます。サ

ービスの実行結果は、Ops Center Automator ユーザーインタフェースに、サービス出力プロパティの値として出力されます。ステップの実行に必要な入力値を格納する入力プロパティと、実行結果を格納する出力プロパティは、部品を通じて定義されます。部品の入力プロパティに直接値を入力できます。または、これらをサービスの入力プロパティまたは変数にリンクすることで、これらに値を渡すことができます。サービス出力プロパティを部品の出力プロパティにリンクすることで、Ops Center Automator のユーザーインタフェースから部品の実行結果を確認できます。この方法でプロパティをリンクし、これらの間で値を受け渡しすることを、プロパティマッピングと呼びます。

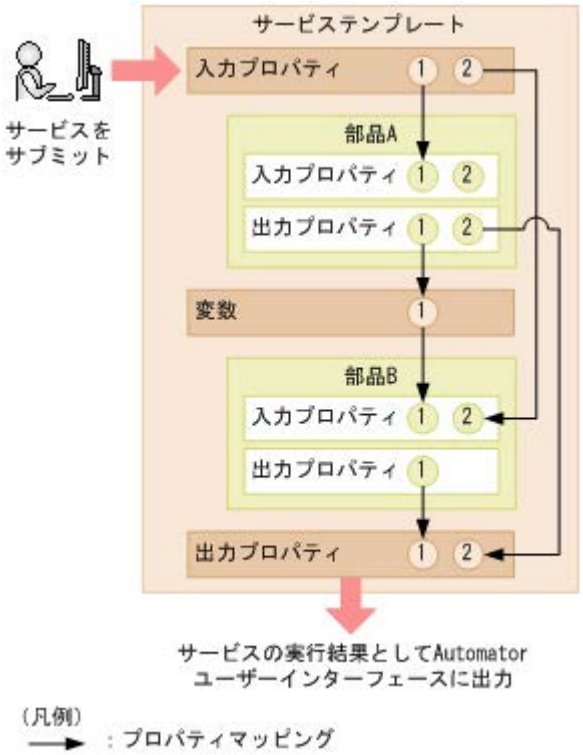
ステップごとにコンポーネントの入力プロパティと出力プロパティを、サービステンプレートの入力および出力プロパティと変数にマッピングする必要があります。サービステンプレートで使用されるコンポーネントの入力プロパティは、入力プロパティまたは変数、もしくはサービステンプレートに関連付けられた、その他のコンポーネントのステップの出力プロパティと関連付ける必要があります。サービステンプレートの入力プロパティはサービスの実行に必要な入力値を格納します。サービステンプレートの出力プロパティはサービスの実行結果を格納します。

部品の入力プロパティの値は、そのプロパティにこれまでに設定されている値、もしくはサービステンプレート用に直接設定した値です。入力プロパティは変数にすることもできます。サービス共有プロパティを入力プロパティに適用することもできます。

出力プロパティの内容はコンポーネントのタイプによって異なります。ステップ実行の結果を出力プロパティとして格納することができます。変数は、コンポーネント間で受け渡しされる値を一時的に保持します。

コンポーネントの入力プロパティと出力プロパティは、[Service Builder Edit] 画面の [プロパティ] タブで設定されます。サービステンプレートの入力プロパティと出力プロパティおよび変数は、[Service Builder Edit] 画面の [プロパティ] タブで設定されます。

次の図に、サービステンプレートのプロパティと対応する部品ステップのプロパティの一般的なマッピングを示します。



この例では、

- サービスの入力プロパティ 1 に入力される値が、部品 A の入力プロパティ 1 に入力されます。
- サービスの入力プロパティ 2 に入力された値が、部品 B の入力プロパティ 2 に入力されます。
- 部品 A がマッピングされていない入力プロパティ 2 には、サービステンプレートが作成または編集されたときに入力された値が割り当てられます。

部品 A の出力プロパティ 1 は、変数 1 にマッピングされており、変数 1 は部品 B の入力プロパティ 1 にマッピングされていることに注目してください。部品 A の出力プロパティ 1 として出力される値は変数 1 に格納され、次に部品 B の入力プロパティ 1 に入力されます。これにより、部品 A の実行結果が部品 B の入力プロパティに渡されるため、この結果を部品の処理に使用できるようになります。このマッピングにより以下の結果が得られます。

- 部品 A の出力プロパティ 2 として出力された部品 A の実行結果（コマンドの標準出力、標準エラー出力、および出力プロパティ）は、サービスの出力プロパティ 2 にも出力されます。これにより、Ops Center Automator ユーザーインタフェースを通じて部品 A の実行結果を確認することができます。
- 部品 B の出力プロパティ 1 として出力された部品 B の実行結果（コマンドの標準出力、標準エラー出力、および出力プロパティ）は、サービスの出力プロパティ 1 にも出力されます。部品 B の実行結果は Ops Center Automator ユーザーインタフェースを通じて確認できます。

前提条件

開発状態のサービステンプレートが存在し、［フロー］ビューにステップが追加されている必要があります。

操作手順

1. ［Service Builder Edit］画面の［フロー］タブで、フロー内のステップをクリックしてハイライト表示します。
   選択したコンポーネント（部品またはほかのサービステンプレート）に関連付けられたプロパティの一覧は、［プロパティ］タブから表示されます。検索ボックスの隣の適切なアイコンをクリックして、入力プロパティまたは出力プロパティを参照できます。検索ボックスでは、検索するテキストを入力して長いリスト内の特定のプロパティを見つけることができます。
2. 入力プロパティのアイコンを選択した状態で、ステップの入力プロパティを入力します。
   - 表の［値］列に詳細を直接入力します。
   - ［鉛筆］アイコンをクリックし、［入力プロパティマッピング］ダイアログを使用して詳細を入力します。
3. 出力プロパティのアイコンを選択した状態で［鉛筆］アイコンをクリックし、［出力プロパティマッピング］ダイアログを使用して出力プロパティの詳細を入力し、［OK］をクリックします。［出力プロパティ追加］ボタンをクリックして新しい出力プロパティを追加したり、［変数追加］ボタンをクリックして変数を追加したりできます。
4. 必要に応じて、引き続き入力および出力プロパティを入力します。
5. 必要な場合は、［サービスプロパティ］チェックボックスをオンにします。このチェックボックスは、選択したプロパティが Modify/Submit ユーザーに対して表示されるかどうかを決定します。

次の作業

サービステンプレートのサービス定義を設定します。

関連参照

- 4.4.4 ［入力プロパティマッピング］または［出力プロパティマッピング］ダイアログ

4.4.4 ［入力プロパティマッピング］または［出力プロパティマッピング］ダイアログ

サービステンプレートに組み込まれているコンポーネントの入力および出力プロパティのマッピングは、［入力プロパティマッピング］または［出力プロパティマッピング］ダイアログで指定できます。

次の表で、ダイアログのフィールド、サブフィールド、およびフィールドグループについて説明します。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 9 ［入力プロパティマッピング］または［出力プロパティマッピング］ダイアログ

フィールド	サブフィールド	説明
［プロパティキー:］	-	入力または出力プロパティキー。
［プロパティ名:］	-	入力または出力プロパティの名前。
［説明:］	-	入力または出力プロパティの説明。
［データ型:］	-	プロパティのデータ型を示します（string, boolean, integer, double, date, password, composite）。どのオプションを選ぶかに応じて、さまざまなオプションが新たに提示されます。 配列を扱う場合、［配列データ］オプションを選択して、データ型を配列として扱うことができます。こうすると、同じタイプのプロパティのセット（要素の数は変動あり）を 1 つのプロパティとして扱うことができ、データマッピングが容易になります。特に、サービスと部品の間でデータのやりとりをする場合に容易になります。
［設定方法:］	-	設定方法は、［プロパティ参照］または［直接入力］のどちらかです。直接入力値を入力するときは、［プロパティ挿入］ボタンをクリックすると、指定したプロパティの値を、その値として挿入できます。
［値:］	-	指定されたプロパティに関連する直接入力値を表示します。

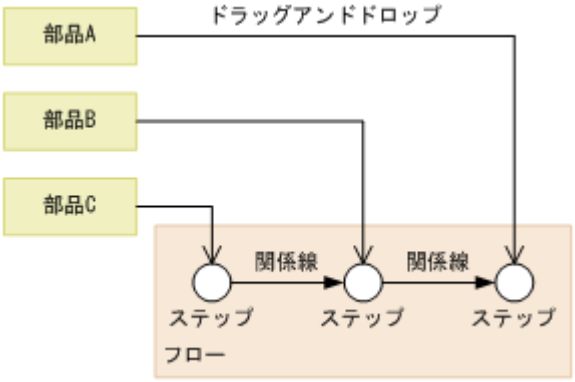
コンポーネントの入力プロパティを指定するときは、現在選択されているコンポーネントの［ステップツリー］の下に表示される部品ステップの一覧を参照できます。

入力プロパティあるいは出力プロパティのどちらを操作しているかに応じ、［入力プロパティの追加］、あるいは［出力プロパティの追加］ボタンをクリックして、入力プロパティまたは出力プロパティを追加できます。また、［変数追加］をクリックして、指定したプロパティグループに変数を追加できます。

4.4.5 実行フローを作成する

サービステンプレートのプロセスフローのマッピングは、［Service Builder Edit］画面の［フロー］タブで実行します。［フロー］ビューを使用してステップの各アイコンを接続し、フローを作成します。

部品を［部品］ビューから［フロー］ビューにドラッグして、フローの各処理単位を作成します。［フロー］ビューにドロップした各部品をステップと呼びます。タスクの実行に必要なステップを配置し、各ステップを関係線で接続することでフローを作成します（次の図を参照）。



フローに含めた 1 つのステップを 2 つ以上のステップに接続できます。同様に、2 つ以上のステップを 1 つのステップに接続することもできます。この場合、接続しているすべてのステップの実行が終了した後にのみ、次のステップが実行されます。階層フロー部品と繰り返し実行部品を使用して、別のフロー内にフローを定義することもできます。

以下の手順に従って、サービステンプレートのステップのフローを作成します。

前提条件
開発状態のサービステンプレートが存在し、[フロー] ビューにステップが追加されている必要があります。

操作手順

1. サービスにより実行されるステップの相対順序に従って、[Service Builder Edit] 画面の [フロー] タブから必要な部品（またはサービス）を [フロー] ビューへドラッグして追加します。
2. フロー内のステップの実行順序を設定するには、最初に実行されるステップに関連付けられているノードをクリックして、ボタンを押したままコネクターの線を、次に実行されるステップまでドラッグし、ボタンを離します。
   接続線が表示され、ステップが接続されたことが示されます。矢印はフローの方向を示します。
3. 引き続き、必要に応じて接続をステップに追加します。
   接続を削除するには、対象の接続線をクリックし、[削除] ボタンをクリックします。

次の作業

入力と出力のプロパティを定義し、マッピングパラメーターを入力します。

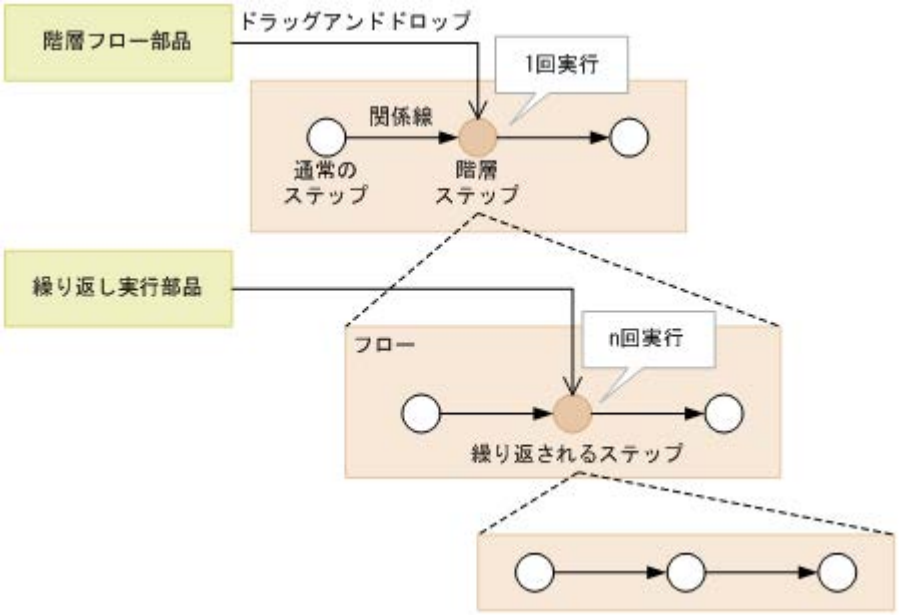
関連タスク

- 4.4.3 ステッププロパティを指定する

4.4.6 フロー階層を作成する

既存のフローの中に別のフローを定義して、サービステンプレート内にフロー階層を作成できます。[フローツリー] ビューに、フロー階層が表示されます。

階層フロー部品を配置して、次の図に示すように繰り返し実行部品を配置することにより複数ステップから構成される処理ユニットを繰り返すことで、フロー階層を作成できます。



次の表は部品の役割とフロー内の各種ステップとの関係を示します。

ドラッグアンドドロップされた部品	ステップのタイプ	ロール
階層フロー部品	階層ステップ	フロー階層を作成する
繰り返し実行部品	繰り返されるステップ	指定されたフローの実行を繰り返します。繰り返されるステップの下位フローに階層を作成する場合には、階層フロー部品を使用する必要があります。コピーアンドペーストしたフローの中に繰り返されるステップがある場合、そのフローに階層を作ろうとするとエラーが発生します。繰り返し実行部品を使用すると、最大 3 レベルのネストループを作成できます。
その他の部品	通常のステップ	通常どおり部品を実行します。

［フローツリー］ビューには、フローのステップ一覧が表示され、フローの最初のステップとしてサービステンプレート名が表示されます。下位レベルは、階層フロー部品または繰り返し実行部品を実行するステップと関連付けられたステップ名で表されます。階層フローを含むサービステンプレートを実行するときは、フローの最上位レベルのステップだけが［タスク詳細］ダイアログに表示されます。下位フローのステップおよび階層フロー部品と繰り返し実行部品のステップは表示されません。

以下の手順に従って、サービステンプレートのステップのフローを作成します。

前提条件
開発状態のサービステンプレートが存在し、［フロー］ビューにステップが追加されている必要があります。

操作手順

1. ［Service Builder Edit］画面の［フロー］タブで、実行する部品を［フロー］ビューにドラッグし、フロー内のほかの部品の実行階層を確立します。

2. 指定したフローの実行を繰り返すには、繰り返し実行部品を選択して適切な場所にドラッグします。
3. 必要に応じてステップへの接続を追加します。
4. （任意）接続を削除するには、対象の接続線をクリックし、［削除］をクリックします。

4.4.7 フローの次ステップの条件分岐を作成する

条件分岐を作成して、特定の条件が満たされた場合に実行されるステップのフローを制御できます。

次ステップの条件分岐は、特定の条件に基づいてフローのステップを実行するのに役立ちます。

サービステンプレートの実行フローを作成するとき、［次ステップ］タブで次ステップの条件を指定します。

実行フローの任意のステップに関連づけられたステップ名と条件設定が表示されます。

- ステップ名：条件が定義されているステップの名前。
- 条件名：条件式名を指定します。デフォルトでは次のステップ名。
- 条件：次のステップを実行するために満たす必要がある条件のタイプ（ALWAYS、IF、OTHER）。OTHER を選択すると、タイプ IF の条件式を持つ項目が必要です。
- 条件式：サービスプロパティに適用される有効な条件式を使用した条件を指定します。
- 説明：条件についての簡単な説明を提供します。

次ステップの条件を追加するには：

操作手順

1. 条件付きフローに含めるステップを画面のフローエリアにドラッグします。
2. 条件付きフローの先行ステップと、条件が発生した場合に実行される次のステップとの間に線を結びます。
3. ［次ステップ］タブで、プルダウンリストから適切な条件（ALWAYS、IF、OTHER）をクリックします。ALWAYS を選択すると、そのステップが通常デフォルトであるのと同様に、ステップが常に実行されます。IF と OTHER は一緒に使用する必要があるため、ほかの選択肢は IF だけです。
4. 鉛筆アイコンをクリックして［実行条件］ダイアログにアクセスすると、条件の値を入力できます。詳細については、有効な演算子を参照してください。

5. 条件値を入力したら、［OK］をクリックします。必要に応じて、次のステップの条件に説明を入力できます。

   終了すると、実行フローの線矢印上の条件付きアイコンによって示されるように条件が設定されます。

   さらに複雑な条件分岐を作成するには、複数の戻り値判定分岐部品を使用して、特定の条件が満たされる場合にあるステップを実行し、条件が満たされない場合に別のステップを実行させます。

4.4.8 ［実行条件］ダイアログ

指定した条件に基づき、フローのどのステップを次に実行するかを指定できます。

次の表で、ダイアログのフィールド、サブフィールド、およびフィールドグループについて説明します。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 10 ［実行条件］ダイアログ

名前	説明	編集可否
［ステップ名］	条件が定義されているステップの名前	不可
［条件名］	条件式名。デフォルトでは次のステップ名。	可
［説明］	評価中の条件についての簡単な説明を提供します。	可
［値］	サービスプロパティに適用される有効な条件式を使用した条件を指定します。	可

有効な演算子は以下になります。

表 11 演算子

シンボル／文字列	意味	ノート
OR	論理 OR	• 演算子には大文字小文字ともに使用可能。 • OR 演算子の両側にはスペースが必要です。 • 論理 AND と一緒には使用できません。複数の論理 OR は一緒に使用できます。
AND	論理 AND	• 演算子には大文字小文字ともに使用可能。 • AND 演算子の両側にはスペースが必要です。 • 論理 OR と一緒には使用できません。複数の論理 AND は一緒に使用できます。
=	等号	• 演算子には文字列または数値が使用可能。 • = 演算子の両側にはスペースが必要です。 • 全角の数字は文字列とみなされます。
!=	不等号	• 演算子には文字列または数値が使用可能。 • != 演算子の両側にはスペースが必要です。 • 全角の数字は文字列とみなされます。
<=	より小さいまたは等しい	• 数値のみ使用可能。

シンボル／文字列	意味	ノート
		• <= 演算子の両側にはスペースが必要です。 • 全角の数字は使用できません。
>=	より大きいまたは等しい	• 数値のみ使用可能。 • >= 演算子の両側にはスペースが必要です。 • 全角の数字は使用できません。
<	より小さい	• 数値のみ使用可能。 • < 演算子の両側にはスペースが必要です。 • 全角の数字は使用できません。
>	より大きい	• 数値のみ使用可能。 • > 演算子の両側にはスペースが必要です。 • 全角の数字は使用できません。
equals	等しい	• 文字列の値のみ使用可能。 • 演算子には大文字小文字ともに使用可能。 • "equals" 演算子の両側にはスペースが必要です。
not equals	不等号	• 文字列の値のみ使用可能。 • 演算子には大文字小文字ともに使用可能。 • "not equals" 演算子の両側にはスペースが必要です。
contains	含む	• 文字列の値のみ使用可能。 • 演算子には大文字小文字ともに使用可能。 • "contains" 演算子の両側にはスペースが必要です。
not contains	含まない	• 文字列の値のみ使用可能。 • 演算子には大文字小文字ともに使用可能。 • "not contains" 演算子の両側にはスペースが必要です。
¥	エスケープ	• 演算子のシンボルを文字列として解釈し区別します。 • 文字列として扱われる条件式の各単語の前にエスケープ演算子を挿入します（例：a ¥not¥equals b）。 • 二重引用符（"）を文字列として扱う場合は、「¥"」と指定します。 • 円記号（¥）を文字列として指定する場合は、「¥¥」と指定します。

4.5 プロパティ設定を指定する

サービステンプレートの入力プロパティと出力プロパティ、サービス共有プロパティ、および変数は、[Service Builder Edit] 画面の［プロパティ］タブで指定します。これらの設定は、特定のテンプレートに関連付けられたさまざまなパラメーターがユーザーに表示される方法に影響します。

［プロパティ］タブから次のタスクを実行できます。

- ［全グループを開く］ または［全グループを閉じる］: プロパティグループの表示を展開または折りたたみます。
- ［追加］: 次のオプションを提供します。
  - ［プロパティグループ］: プロパティの表示を整理するためにプロパティグループを追加します。
  - ［入力プロパティ］: 選択したサービステンプレートの新しい入力プロパティを作成します。
  - ［出力プロパティ］: 選択したサービステンプレートの新しい出力プロパティを作成します。
  - ［変数］: 選択したサービステンプレートの新しい変数を作成します。
  - ［サービス共有プロパティ］: サービス共有プロパティを追加します。
- ［編集］: 選択したプロパティの詳細を編集します。
- ［削除］: 選択したプロパティを削除します。
- ［プレビュー］: ［サービス作成］画面、［サービス実行］画面、または［タスク詳細］画面から、プロパティの使用法をプレビューします。
- ［その他の操作］: 次のオプションを提供します。
  - ［可視性を変更］: ［サービス作成］または［サービス作成とサービス実行］で、これらの項目がユーザーに対して表示されるかどうかを指定します。
  - ［表示設定を変更］: 表示設定を指定して、ユーザーが設定を変更可能（[変更可能]）、読み取り専用（[変更不可]）、表示（[表示]）、または非表示（[非表示]）にします。

一部のケースでは、[サービス詳細説明表示] ダイアログおよびサービスに関連付けられたその他の概要に表示されるデフォルト値のアイコンやテキストおよびグラフィックを変更してサービステンプレートをカスタマイズしたほうがよい場合があります。

4.5.1 サービス共有プロパティを選択する

サービステンプレートにサービス共有プロパティを追加して、よく使用する（定義済みの）機能を実行できます。

前提条件
開発状態のサービステンプレートが存在し、[フロー] ビューにステップが追加されている必要があります。

操作手順

1. [Service Builder Edit] 画面の［プロパティ］タブにアクセスします。
2. ［追加］プルダウンメニューから［サービス共有プロパティ］オプションを選択します。
   [サービス共有プロパティ選択] ダイアログが表示されます。現在のサービステンプレートに、特定のプロパティグループに割り当てることができるサービス共有プロパティのリストが提供されています。
3. 提供されたリストから、サービステンプレートに追加するサービス共有プロパティを選択し、［OK］をクリックします。
   サービステンプレートに選択したサービス共有プロパティが追加されます。

次の作業

- 選択したサービス共有プロパティの値を設定するには、[サービスの入力プロパティ編集] ダイアログにアクセスします。
- また、[追加] プルダウンメニューから [プロパティグループ] オプションを選択して適切な属性を指定することで、特定のプロパティグループにサービス共有プロパティを追加できます。

関連参照

- 4.5.2 [サービス共有プロパティ選択] ダイアログ

4.5.2 [サービス共有プロパティ選択] ダイアログ

[プロパティグループ] のサービス共有プロパティは、[サービス共有プロパティ選択] ダイアログで選択します。

次の表は、[サービス共有プロパティ選択] ダイアログのフィールド、サブフィールド、およびフィールドグループについての説明です。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

表 12 [サービス共有プロパティ選択] ダイアログ

フィールド	サブフィールド	説明
[プロパティ名:]	-	選択したプロパティに割り当てられている表示名。
[プロパティキー:]	-	サービス共有プロパティに関連付けられたキー名。
[説明:]	-	サービス共有プロパティが実行する機能の説明。
[プロパティグループ:]	-	サービス共有プロパティを割り当てるプロパティグループを指定します。

[編集] ボタンをクリックすることで、サービス共有プロパティに値を設定できます。

関連タスク

- 4.5.1 サービス共有プロパティを選択する

4.5.3 [参照プロパティ選択] ダイアログ

[参照プロパティ選択] ダイアログより参照プロパティを選択します。

次の表は、[参照プロパティ選択] ダイアログのフィールド、サブフィールド、およびフィールドグループについての説明です。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

表 13 [参照プロパティ選択] ダイアログ

フィールド	サブフィールド	説明
[ステップツリーエリア]	-	テンプレートに定義されたステップの一覧を表示します。ステップツリーエリアの情報は、以下のアイコンでハイライト表示されます。 • 予約プロパティ • サービステンプレート • サービス部品

フィールド	サブフィールド	説明
		• 階層フロー部品 • 繰り返し実行部品 • その他
[プロパティリスト]	-	ステップツリーエリアで選択されたグループに定義されたプロパティの、プロパティ名とプロパティキーを表示します。初期表示では、編集中のプロパティのステップよりも前に実行されるステップのみ表示されます。
[すべてのステップを表示]	-	ON の場合、テンプレートに定義されたすべてのステップが表示されます。初期表示では OFF になっています。編集中のプロパティが出力プロパティの場合は、すべてのステップは表示されません。
[予約プロパティ：]	-	繰り返し実行部品がステップツリーエリアで選択されている場合、部品に関連した予約プロパティが表示されます。この項目は、繰り返し実行部品がステップツリーエリアで選択されている場合にのみ表示されます。

選択可能な、繰り返し実行入力値（reserved.loop.input、reserved.loop.input***N***）と、繰り返し実行ループインデックス （reserved.loop.index、reserved.loop.index***N***）はプルダウンに表示され、一行にまとめて表示されます。繰り返し実行入力値と繰り返し実行ループインデックス、それに対応する繰り返し実行部品の関係をわかりやすくするため、プロパティ名に対応する繰り返し実行部品のステップ名が括弧で表示されます。

以下に予約プロパティの説明を示します。

表 14 予約プロパティ

プロパティ	表示条件	表示名とフォーマット
reserved.loop.input	このダイアログを開始した、部品から見たベースステップより 1 つ上のレベルにある繰り返し実行部品をステップツリーエリアで選択する場合	1 個上位の繰り返し実行入力値 (*対応するステップ名*)
reserved.loop.index		1 個上位の繰り返し実行の回次(*対応するステップ名*)
reserved.loop.input***N***	このダイアログを開始した、部品から見たベースステップより ***N*** レベル上にある繰り返し実行部品をステップツリーエリアで選択する場合	***N*** 個上位の繰り返し実行入力値 (*対応するステップ名*)
reserved.loop.index***N***		***N*** 個上位の繰り返し実行の回次 (*対応するステップ名*)

4.5.4 入力プロパティを追加する

サービステンプレートに関連付けられたプロパティグループに入力プロパティを指定できます。

前提条件

開発状態のサービステンプレートが存在し、[フロー] ビューにステップが追加されている必要があります。

操作手順

1. 次のどちらかの方法で、サービステンプレートに関連付けられたプロパティグループに入力プロパティを追加します。

- ［Service Builder Edit］画面の［プロパティ］タブで、［追加］プルダウンメニューから［入力プロパティ］オプションを選択します。
- ［フロー］タブの［ステッププロパティ］エリアにある［値］フィールドの［鉛筆］アイコンをクリックします。入力プロパティを追加できる［入力プロパティマッピング］ダイアログが表示されます。［入力プロパティの追加］ボタンをクリックします。

2. ［サービスの入力プロパティ作成］ダイアログが表示されます。
3. 入力プロパティに関連する詳細を入力します。
4. ［OK］をクリックして、指定した入力プロパティの詳細を保存します。
   ここで指定した入力プロパティは、後で入力プロパティ一覧に反映されます。

関連参照

- 4.5.5 ［サービスの入力プロパティ作成］または［サービスの入力プロパティ編集］ダイアログ

4.5.5 ［サービスの入力プロパティ作成］または［サービスの入力プロパティ編集］ダイアログ

次の表は、［サービスの入力プロパティ作成］または［サービスの入力プロパティ編集］ダイアログのフィールド、サブフィールド、およびフィールドグループについての説明です。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 15 プロパティ定義フィールドグループ

フィールド	サブフィールド	説明
［プロパティキー:］*	-	入力プロパティキー。
［プロパティ名:］*	-	入力プロパティの名前。
［説明:］	-	入力プロパティの説明。
［プロパティグループ:］	-	プロパティが所属するプロパティグループを選択します。［新しいプロパティグループを作成］を選択して、新しいプロパティグループを作成することもできます。
［可視性:］	-	入力プロパティを［サービス作成］および［サービス実行］画面の両方に表示するのか、［サービス作成］画面だけに表示するのかを選びます。
［表示設定:］	-	入力プロパティの表示設定を指定します。選択肢は次のとおりです。 • 変更可能 • 変更不可 • 非表示
［スコープ:］	-	サービス共有プロパティを有効にしてリリースプロセス後にサービスをサービス部品として追加します。
［必須区分:］	-	このチェックボックスが選択された場合、プロパティが必須であることを示します。
［データ型:］	-	プロパティのデータ型を選択します（string、boolean、integer、double、date、password、composite）。どのオプションを選択するかによって、データ入力の際の制約を指定する様々なオプションが提示されます。

フィールド	サブフィールド	説明
		配列を扱う場合、［配列データ］オプションを選択して、データ型を配列として扱うことができます。こうすると、同じタイプのプロパティのセット（要素の数は変動あり）をひとつのプロパティとして扱うことができ、データマッピングが容易になります。特に、サービスと部品の間でデータのやりとりをする場合に容易になります。
［データ形式:］	-	データ形式を選択します。 • application/json • application/javascript • application/xml • text/html • text/plain • text/csv • application/octet-stream
［ドメインタイプ:］	-	リストからドメインタイプを選択します。または、［新規ドメインタイプ作成］をクリックし、［Create Domain Type Definition］ダイアログに詳細を入力して、新しいドメインタイプを追加します。このオプションは、データ型に composite を選択し、データ形式に application/json を選択すると使用できます。

アスタリスク（*）が付いたフィールドは必須です。

表 16 制約条件フィールドグループ

フィールド	サブフィールド	説明
［最小値:］	-	最小値を指定します。この入力フィールドは、データ型に integer、double、または date を選択すると表示されます。
［最大値:］	-	最大値を指定します。この入力フィールドは、データ型に integer、double、または date を選択すると表示されます。
［最小長:］	-	プロパティの最小の長さを指定します。この入力フィールドは、データ型に string または password を選択すると表示されます。
［最大長:］	-	プロパティの最大の長さを指定します。この入力フィールドは、データ型に string または password を選択すると表示されます。
［入力文字制限:］	-	正規表現を使用して、許可する文字を指定します。この入力フィールドは、データ型に string または password を選択すると表示されます。 例：^[0-9a-zA-Z¥.¥-]*$
［最小配列長:］	-	配列の要素の最小長を指定します。
［最大配列長:］	-	配列の要素の最大長を指定します。
［バリデーションスクリプト:］	-	関連する Javascript コードに基づいてプロパティを検証するスクリプトです。

表 17 値と表現形式フィールドグループ

フィールド	サブフィールド	説明
［表現形式:］	-	プロパティの表現形式を選択します。データ型に応じて選択可能な表現形式がリストに表示されます。
［デフォルト値:］	-	プロパティのデフォルト値が［true］か［false］かを指定します。［配列データ］オプションを指定する場合、デフォルト値は、角括弧で囲まれたコンマ区切りの文字列値で入力する必要があります。 例：["1","2","3"]
［データソース:］	-	データが［静的］であるか、［動的］かつ外部リソースプロバイダから取得されたものかどうかを指定します。
［リスト表示値:］	-	プロパティのデータソースを静的に取得する場合（［データソース］に［静的］オプションを選択した場合）、リスト表示値を指定します。
［外部リソース:］	-	プロパティのデータソースを動的に取得する場合（［データソース］に［動的］オプションを選択した場合）、外部リソースプロバイダを指定します。 リストで、外部リソースプロバイダを追加、編集、アップロード、または削除することもできます。
［追加パス:］	-	リクエスト URL の追加パス部分を指定します。不要な場合は、空のままにしておきます。追加パスは、次に示すように、URL 内で外部リソースプロバイダ ID の後に続くパスのことです。 /Automation/v1/objects/ExternalResources/<*外部リソースプロバイダ ID* >/<*追加パス*>?<*クエリパラメタ*>
［クエリパラメタ:］	-	外部リソースプロバイダのためのクエリパラメタを指定します。`serviceID` プロパティと `serviceTemplateID` プロパティは自動的に追加されます。{$ref:*keyName*}を指定すると、同じプロパティグループ内の他のプロパティの値を参照できます。JSON 値の場合、{$ref:*keyName#json path*}を指定できます。
［名前フィールド:］	-	リストに表示されるラベルとして使用するフィールド名を指定します。省略した場合、name フィールドが使用されます。
［値フィールド:］	-	リストに表示されるプロパティの値として使用するフィールド値を指定します。省略した場合、instance ID フィールドが使用されます。
［表示条件:］	-	指定した条件を満たす場合、プロパティを表示します。
［活性化条件:］	-	指定した条件を満たす場合、プロパティを活性化します。

入力プロパティをチェックするための検証スクリプトを作成する

提供されている検証オプションが目的にそぐわない場合には、スクリプトを作成して必要なチェックを行わせることができます。次に示すのは、ユーザーの入力値が許される最大値である 1024 以下の数字であることをチェックするために JavaScript で書いた検証スクリプトです。

```
function(propertyValue, lang, displayType){
  var jObject = JSON.parse(propertyValue.value);
  if(displayType == "config"){
    if( jObject.luSize > 10){
      return "lu size should be under 10"
    }
    if( jObject.blockSize > 2){
      return "block size should be under 2";
```

```
    }
  }

 return
}
```

次の表は入力プロパティ用の検証スクリプトを示します。

名前	説明

スクリプトフォーマット	function (arg1, arg2, arg3) {
	//コード
	}

検証スクリプトの引数	arg1: 文字列形式のプロパティ値 arg2: ロケール文字列。例：ja、 en arg3: スクリプト動作中の操作情報（タスク作成操作：exec、プロパティの編集操作：config）
検証スクリプトの戻り値※	成功： undefined、null 失敗： 配列または文字列形式のエラーメッセージ

注※　値が数値ではないか、または指定最大値よりも大きい場合には、ユーザーインタフェースを通じてメッセージが出力されます。

入力プロパティに外部リソースデータを使用する

入力プロパティの表現形式として"Selection"を指定すると、メニューのリスト値として外部リソースデータを使用できます。外部リソースデータを設定およびアクセスするためには、Service Builder を使用して外部リソースプロバイダを作成します。

Service Builder を使用して外部リソースプロバイダを作成するには

新しい外部リソースプロバイダを作成するには、リストの下部にある［新規リソースプロバイダ作成］をクリックします。リストから既存の外部リソースプロバイダを指定することもできます。

［外部リソースプロバイダ作成］ダイアログに次の情報を入力します。

- 名前：外部リソースプロバイダの名前を指定します。
- バージョン：バージョン番号を指定します。
- コンテントタイプ：application/json または text/csv を選択します。
- スキーマ ID：外部リソースプロバイダに対応するドメインタイプのスキーマ ID を指定します。サービスおよび部品の入力プロパティ編集ダイアログのプロパティで、外部リソースプロバイダを選択するのに役立ちます。選択したドメインタイプと同じスキーマ ID を持つものは、外部リソースプロバイダのリストで、青でハイライト表示されます。
- 説明：外部リソースプロバイダの説明を指定します。
- タイプ：Javascript、スクリプト、コマンドライン、ファイルのどれかを選択します。選択したタイプに応じて、必要な情報をフィールドに入力します。

［タイプ］オプションに Javascript を指定する場合

次の表に、外部リソースプロバイダの Javascript についての仕様を示します。

名前	説明
スクリプトフォーマット	function fn (requestPath, queryParamMap, properties) { // code }
スクリプトの引数	requestPath: ［サービスの入力プロパティ作成］または［サービスの入力プロパティ編集］ダイアログで指定した追加パスの値。 queryParamMap: ［サービスの入力プロパティ作成］または［サービスの入力プロパティ編集］ダイアログでクエリパラメタとして指定したキーと値のマップを含む JSON オブジェクト。 properties: 外部リソースプロバイダに関連するサービス共有プロパティと予約プロパティを含む JSON オブジェクト。予約プロパティは次のとおりです。 ・ reserved.external.hcmds.dir ・ reserved.external.path ・ reserved.external.query ・ reserved.external.resource.dir ・ reserved.external.userName プロパティの値は、プロパティ["*property key*"]で取得できます。
スクリプトの戻り値	プルダウンにリストされる JSON オブジェクトの配列を返します。返される配列は、"data"というプロパティに設定される必要があります。

スクリプトでは、次のユーティリティ関数のいずれか 1 つを使用できます。

- 組み込みの CM-REST メソッド
- env 関数
- auto util ライブラリ

メモ　組み込みの CM-REST メソッドと env 関数を含む JavaScript Plug-in for Configuration Manager REST API は、将来のバージョンでサポート終了予定のため、外部リソースプロバイダで Configuration Manager REST API を実行するときは、auto util ライブラリを使用することを推奨します。組み込みの CM-REST メソッドおよび env 関数から auto util ライブラリに移行するには、このトピックの「auto util ライブラリを使用する」を参照してください。

組み込みの CM-REST メソッドを使用する

次に示す、JavaScript Plug-in for Configuration Manager REST API で利用できるモジュールを使用できます。

- ConfigurationManager._0x_xx_xx.api
- ConfigurationManager._0x_xx_xx.model
- ConfigurationManager._0x_xx_xx.enum
- ConfigurationManager._0x_xx_xx.lib

メモ　「_0x_xx_xx」は、JavaScript Plug-in for Configuration Manager REST API のバージョンに対応しています。バージョンが 01.51.01 の場合、「_01_51_01」となります。

env 関数を使用する

env 関数を使用してコンテキスト情報を取得できます。次の表に env オブジェクトの関数の仕様を示します。

機能	説明
env.getWebServiceConnections(category)	Web サービスの接続先のリストを取得します。 入力パラメーター： category:Web サービス接続先のカテゴリの名称 戻り値： 入力パラメーターとして指定されたカテゴリ名を持つ Web サービス接続先の JSON オブジェクトのリスト。外部リソースプロバイダがサービスによって呼び出されると、サービスによるアクセスが可能な Web サービス接続先が取得されます。その際に、サービス、サービスグループ、インフラストラクチャーグループ、および Web サービス接続先の定義済みの関連性が考慮されます。
env.setWebServiceConnection(apiObject, category, name, basePath)	apiObject に Web サービス接続先を設定します。指定した apiObject により、指定した Web サービス接続先が外部接続に使用されます。 入力パラメーター： ・ apiObject:組み込みの CM-REST モジュールの ObjectsApi のインスタンス ・ category:Web サービス接続先のカテゴリの名称 ・ name:Web サービス接続先の名前 ・ basePath:"/"で始まる URL のポート番号が続くパス 戻り値： 戻り値はありません。
env.setHTTPProxy(apiObject, host, port, authenticationSchema, userName, password)	Web サービス接続先を使用しない場合は、apiObject に HTTP プロキシを設定します。 入力パラメーター： ・ apiObject:組み込みの CM-REST モジュールの ObjectsApi のインスタンス ・ host:プロキシサーバのホスト名または URL ・ port:プロキシサーバのポート番号 ・ authenticationSchema: basic または digest ・ userName:認証するユーザー名 ・ password:認証するパスワード 戻り値： 戻り値はありません。
env.getServiceTemplate()	外部リソースプロバイダを呼び出して、サービステンプレートのオブジェクトを取得します。 入力パラメーター： 入力パラメーターはありません。 戻り値： サービステンプレートオブジェクト
env.getService()	外部リソースプロバイダを呼び出して、サービスのオブジェクトを取得します。 入力パラメーター： 入力パラメーターはありません。

機能	説明
	戻り値： サービスオブジェクト

auto util ライブラリを使用する

auto util ライブラリを使用できます。詳細については、「B.16 JavaScript 実行部品」の「auto util ライブラリ」のセクションを確認してください。

Configuration Manager REST API を実行する方法として、組み込みの CM-REST メソッドと env 関数を使用するコードから、auto util ライブラリを使用するコードに移行する場合のサンプルコードを次に示します。

以下に、移行前の env 関数と組み込みの CM-REST メソッドのサンプルコードを示します。

プール情報を取得する

```
function fn(requestPath, queryParamMap, properties) {
  /** This is sample code that calls the Configuration Manager REST API.
*/
  /** Replace <VERSION> with the available version of the JavaScript
Plug-in for Configuration Manager REST API. (e.g., put "_01_62_00" for
the version of 01.62.00) */

  //Step 1. Instantiate the API Client.
  var client = new ConfigurationManager._02_80_00.api.ObjectsApi();

  //Step 2. Specify the User Credentials. There are two methods for
specifying user credentials. You can use either method, but the first
one is recommended.
  //Specify credentials by using the Web Service Connection.
  //Get accessible Web Service Connections by specifying a category
name, and specify it. You do not need to specify credentials in a script.
  var wsc = env.getWebServiceConnections("ConfigurationManager");
  env.setWebServiceConnection(client, wsc[0].productName, wsc[0].name, "/
ConfigurationManager");

  //Step 3. Get Session
  //You can pass arguments by using the plug-in input properties or you
can specify them directly in a script. For example, you can pass the
device ID through the query parameters (queryParamMap).
  var argSessionsPost = new
ConfigurationManager._02_80_00.argDef.ObjectsApi.v1.objects.storages.stor
ageDeviceID.sessions.post();
  argSessionsPost.setStorageDeviceID(queryParamMap.deviceId);
  var responseBody =
client.v1.objects.storages.storageDeviceID.sessions.post(argSessionsPost)
;
  var token = responseBody.getToken();
  var sId = responseBody.getSessionId();

  //Step 4. Call the API that is associated with your use case based on
the session obtained in Step 3.
  var argPoolsGet = new
ConfigurationManager._02_80_00.argDef.ObjectsApi.v1.objects.storages.stor
ageDeviceID.pools.get();
  argPoolsGet.setStorageDeviceID(queryParamMap.deviceId);
  argPoolsGet.setPoolType("DP");
  client.getApiClient().setApiKeyPrefix("Session");
  client.getApiClient().setApiKey(token);
  var pools =
client.v1.objects.storages.storageDeviceID.pools.get(argPoolsGet);

  //Step 5. Make an array containing the required information.
  //An array to be returned has to be set to a property named "data".
  var ret = { "data": [] };
  var p;
```

```
  for (var i = 0; i < pools.getData().length; i++) {
    p = pools.getData()[i];
    ret.data.push({
      "Pool ID": p.getPoolId(),
      "Pool Type": p.getPoolType(),
      "Num of LDEVs": p.getNumOfLdevs()
    });
  }

  //Final Step. Discard Session.
  var argSessionIdDelete = new
ConfigurationManager._02_80_00.argDef.ObjectsApi.v1.objects.storages.stor
ageDeviceID.sessions.sessionId.delete();
  argSessionIdDelete.setStorageDeviceID(queryParamMap.deviceId);
  argSessionIdDelete.setSessionId(String(sId));

client.v1.objects.storages.storageDeviceID.sessions.sessionId.delete(argS
essionIdDelete);

  return ret;

}
```

以下に、移行後の auto util ライブラリのサンプルコードを示します。

プール情報を取得する

```
function fn(requestPath, queryParamMap, properties) {
  /** This is sample code that calls the Configuration Manager REST API.
*/

  //Step 1. Generate a method to run REST API to Configuration Manager.
  var configurationManagerCall = function(request) {
    var respBody = null;
    auto.util.http.handleCall(auto.util.storage.restCall, request,
      function(resp, req) {
        respBody = auto.util.parseJson(resp.responseBody);
      }, function(resp, req) {
        auto.util.http.defaultErrorHandler(null, req, resp);
      }, function(err, req) {
        auto.util.http.defaultErrorHandler(err, req);
      }, auto.util.storage.retrySettings
    );
    return respBody;
  };

  //Step 2. Get accessible Web Service Connections by specifying a
category name.
  var wsc =
auto.util.env.getWebServiceConnections("ConfigurationManager");

  //Step 3. Get Session
  //You can pass arguments by using the plug-in input properties or you
can specify them directly in a script. For example, you can pass the
device ID through the query parameters (queryParamMap).
  var request = {
    "requestMethod": "POST",
    "requestUrl": "/ConfigurationManager/v1/objects/storages/" +
queryParamMap.deviceId + "/sessions",
    "requestHeaders": auto.util.http.toRawHeader({}),
    "authScheme": "basic",
    "connectionName": wsc[0].name,
    "productName": wsc[0].productName
  };
  var respBody = configurationManagerCall(request);
  var token = respBody.token;
  var sId = respBody.sessionId;

  //Step 4. Call the API that is associated with your use case based on
the session obtained in Step 3.
//When specifying the Authorization header for the request in the user
program, specify "none" for authScheme.
```

```
  var headers = {
    "Authorization": "Session " + token,
  };
  var queryMap = {
    "poolType" : "DP"
  };
  var queryStr = auto.util.http.buildQuery(queryMap);
  request = {
    "requestMethod": "GET",
    "requestUrl": "/ConfigurationManager/v1/objects/storages/" +
queryParamMap.deviceId + "/pools" + queryStr,
    "requestHeaders": auto.util.http.toRawHeader(headers),
    "authScheme": "none",
    "connectionName": wsc[0].name,
    "productName": wsc[0].productName
  };
  var pools = configurationManagerCall(request);

  //Step 5. Make an array containing the required information.
  //An array to be returned has to be set to a property named "data".
  var ret = { "data": [] };
  var p;
  for (var i = 0; i < pools.data.length; i++) {
    p = pools.data[i];
    ret.data.push({
      "Pool ID": p.poolId,
      "Pool Type": p.poolType,
      "Num of LDEVs": p.numOfLdevs
    });
  }

  //Final Step. Discard Session.
  request = {
    "requestMethod": "DELETE",
    "requestUrl": "/ConfigurationManager/v1/objects/storages/" +
queryParamMap.deviceId + "/sessions/" + sId,
    "requestHeaders": auto.util.http.toRawHeader(headers),
    "authScheme": "none",
    "connectionName": wsc[0].name,
    "productName": wsc[0].productName
  };
  configurationManagerCall(request);

  return ret;

}
```

［タイプ］オプションにスクリプトを指定する場合

スクリプトタイプとして、Python がサポートされています。スクリプトを実行する Python インタプリタへのパスを指定し、スクリプトを編集します。サポートされている Python のバージョンは、3.x シリーズです。この外部リソースプロバイダをクラスタ環境で使用するには、実行系および待機系システムの両方に Python インタプリタがインストールされている必要があります。この外部リソースプロバイダは Python の仮想環境をサポートしていません。

スクリプトから参照できる環境変数

以下の表は、スクリプトで使用できる環境変数を示しています。os.environ[キー*名*]または os.environ.get(キー*名*)の形式で、以下の環境変数の値を取得できます。

環境変数	説明	フォーマット
REQUEST_PATH	［サービスの入力プロパティ作成］または［サービスの入力プロパティ編集］ダイアログで［追加パス］として指定された情報	文字列 *|外部リソースプロバイダ ID|追加パスとして指定された値*

環境変数	説明	フォーマット
QUERY_PARAM_MAP	［サービスの入力プロパティ作成］または［サービスの入力プロパティ編集］ダイアログで［クエリパラメタ］として指定された情報	JSON 形式 {*プロパティ名:値*, ...}
SERVICE_TEMPLATE_ID	Python 実行部品に属するサービステンプレート ID	数値
SERVICE_ID	Python 実行部品を実行するサービス ID	数値
SERVICE_TEMPLATE	Python 実行部品に属するサービステンプレートの情報	JSON 形式 {*サービステンプレート属性:値, ...*}
SERVICE	Python 実行部品を実行するサービスの情報	JSON 形式 {*サービス属性:値*, ...}
WEB_SERVICE_CONNECTIONS	Web サービス接続の設定情報。クエリパラメタに指定されたクエリパラメタ "__webServiceConnectionCategory__"および "__webServiceConnectionName__"に対応した情報が格納されます。格納される情報はクエリパラメタの指定条件によって異なります（下表参照）。	JSON 形式 [{*Web サービス接続属性:値, ...*}, ...]

クエリパラメタ		参照情報
__webServiceConnectionCategory__	__webServiceConnectionName__	
指定あり	指定あり	指定したカテゴリと名前に一致する Web サービス接続情報
指定あり	指定なし	指定したカテゴリに一致する Web サービス接続情報
指定なし	指定あり	なし
指定なし	指定なし	なし

［タイプ］オプションにコマンドラインを指定する場合

実行するコマンドラインを入力します。

- コマンドからの戻り値が 0 でない場合には、エラーとなります。
- 標準出力の上限は 30MB で、この上限を超える場合にはエラーとなります。
- スクリプトファイルを指定する場合には、カレントパスが指定されていないため、絶対パスを使用してください。
- サービス共有プロパティと予約プロパティはコマンド行に含められます。サービス共有プロパティまたは予約プロパティを含める場合には、プロパティキーを"${"および"}"で囲みます。
- 標準出力の読み出し時の文字セットは、システムおよび関連するユーザーの文字セットであると仮定します（例えば、ロケールが日本語に設定されている Windows の場合は MS-932 で、Linux の場合にはユーザーは start コマンドを実行したユーザーになります）。

- コマンドラインで、環境変数の値を参照できます。参照できる環境変数は、［タイプ］オプションにスクリプトを指定する場合と同じです。次の表に示すように、環境変数の値を取得する方法は、スクリプト言語によって異なります。

項目	環境変数の値を取得する方法
コマンドスクリプト（Windows）	%*キー名*%
シェルスクリプト（Linux）	$*キー名*
PowerShell スクリプト	$env:*キー名*

次の例では powershell のようなプログラムを通じてホスト名を取得し、リストに表示する方法を示します。

```
コマンドライン：
powershell.exe ¥"& '${reserved.external.resource.dir}¥¥getHosts.ps1' $
{reserved.external.hcmds.dir} ${reserved.external.userName}¥"

コマンドラインの出力:
name,instanceID
host1,123
host2,124
host3,125
host4,126
host5,127

サービスの[Config/Submit]画面に表示されるリスト項目:
host1
host2
host3
host4
host5
```

［タイプ］オプションにファイルを指定する場合

入力ファイルのパスを指定します。サービス共有プロパティと予約プロパティはファイルのパスに含めることができます。サービス共有プロパティまたは予約プロパティを含める場合には、プロパティキーを"${"および"}"で囲みます。

次の例では、ホスト情報を JSON ファイルから取得し、他のアプリケーションを使って出力しています。

```
ファイル：
${reserved.external.resource.dir}¥¥vm.json

vm.json の内容：
{
"data" :[ {
"instanceID" :127,
"name" :"test1"
}, {
"instanceID" :128,
"name" :"test2"
} ]
}

サービスの[Config/Submit]画面に表示されるリスト項目：
test1
test2
```

外部リソースプロバイダにファイルをアップロードするには

外部リソースプロバイダのプルダウンリストにある［Upload］ボタンをクリックすることで、外部リソースプロバイダにファイルをアップロードできます。A .zip アーカイブファイルは更新できます。ファイルの更新後、ファイルパスとしてコマンドラインに相対パス名で更新したファイルを指定できます。

外部リソースプロバイダを削除する

外部リソースプロバイダのプルダウンリストにある［削除］ボタンをクリックすることで、外部リソースプロバイダを削除できます。［削除］ボタンをクリックすると、削除を確認するダイアログが表示されます。ダイアログには、削除しようとしている外部リソースプロバイダを使用している、サービステンプレートと部品の一覧が表示されます。関連するサービステンプレートと部品がないことを確認し、［OK］ボタンをクリックすると、外部リソースが削除されます。サービステンプレートが使用している外部リソースプロバイダを削除すると、サービステンプレートの外部リソースプロバイダが機能しなくなります。

関連タスク

- 4.5.4 入力プロパティを追加する

4.5.6 ［Create Domain Type Definition］または［Edit Domain Type Definition］ダイアログ

［サービスの出力プロパティ編集］、［サービスの出力プロパティ作成］、［サービスの入力プロパティ編集］、［サービスの入力プロパティ作成］ダイアログから利用できる［ドメインタイプ］オプションを選択し、ドメインタイプの隣にある加算記号（＋）またはレンチのアイコンをクリックすることで、選択したドメインタイプのスキーマを編集できます。

次の表は、［Edit Domain Type Definition］ダイアログのフィールド、サブフィールド、およびフィールドグループについての説明です。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 18 ［Edit Domain Type Definition］ダイアログ

フィールド	サブフィールド	説明
［ドメインタイプ名称］*	-	ドメインタイプの名前を指定します。
［ドメインタイプ ID］	-	ドメインタイプの ID（アドレス）を指定します。
［ドメインタイプスキーマ］	-	ドメインタイプのスキーマを指定します。 形式は、JSON スキーマに準拠します。(http://json-schema.org/)
［雛形生成（ボタン）］	-	ドメインタイプスキーマに基づいて GUI メタ情報を生成します。基本構造はドメインタイプ ID と同じであるため、一から GUI メタ情報を入力する必要はありません。
［GUI メタ情報 ID］	-	GUI メタ情報 ID を指定します。
［GUI メタ情報］	-	GUI メタ情報を指定します。

アスタリスク（*）が付いたフィールドは必須です。

次の表に、GUI メタ情報のプロパティ定義のための属性を示します。

プロパティキー	説明
displayName	プロパティの名称。
description	プロパティの説明。
presentation	プロパティタイプの表現形式。"input"、"textarea"、"url"、"select"、"radio"、"check box"、"spinbox"、"capacity"、"capacityInKB"、"capacityInMB"、"capacityInGB"、"capacityInTB"、"capacityiB"、"capacityInKiB"、"capacityInMiB"、"capacityInGiB"、"capacityInTiB"、"datePicker"、"hex"または"file"。
permission	プロパティのパーミッション。"locked"、"unlocked"または"hidden"。
visibility	プロパティの可視性。"exec"または"config"。
required	必須区分。true または false。
pattern	プロパティ値の利用可能な正規表現パターン。
validationScript	プロパティ値のバリデーションスクリプト。
showIf	プロパティの表示条件。
enableIf	プロパティの活性化条件。
enum	配列を指定します。JSON オブジェクトを定義します。
enumDataSource	外部リソースプロバイダを指定します。以下は一例です。 `"enumDataSource":{ "url":"/Automation/v1/objects/ExternalResources/1eb39858-48b5-4d3a-b82a-9af5c91af8c4", "contentType":"application/json", "nameField":"hostGroupName", "valueField":"id" }`
contentType	composite タイプのデータ形式。 "application/json"、"application/javascript"、"application/xml"、"text/html"、"text/plain"、"text/csv"、"application/octet-stream"。
hidden	表の列を表示または非表示にします。 true：デフォルトで列を非表示にします。 false：デフォルトで列を表示にします。

組み込みの Allocate Volume サービスの Volume Setting を表す、ドメインタイプスキーマと GUI メタ情報の例を以下に示します。

ドメインタイプスキーマ

```
{
  "type":"object",
  "properties":{
    "values":{
      "type":"array",
      "title":"Volume Settings",
      "minItems":1,
      "maxItems":10,
      "items":{
        "type":"object",
```

```
        "properties":{
          "usage":{
            "type":"string",
            "minLength":1,
            "maxLength":64,
            "pattern":"^[A-Za-z0-9 ~!@#¥¥$%¥¥^&()_¥¥+¥¥-=¥¥{¥¥}¥¥[¥¥]'¥¥.`]*
$",
            "default":""
          },
          "numberOfVolumes":{
            "type":"integer",
            "minimum":1,
            "maximum":500,
            "default":2
          },
          "capacity":{
            "type":"integer",
            "minimum":10,
            "maximum":200,
            "default":50
          },
          "storageProfile":{
            "type":"string",
            "enum":[
              "Gold Write",
              "Silver Write"
            ],
            "default":"Gold Write"
          },
          "ldevLabel":{
            "type":"string",
            "minLength":0,
            "maxLength":64,
            "pattern":"^[A-Za-z0-9 ~!@#¥¥$%¥¥^&¥¥*()_¥¥+¥¥-=¥¥{¥¥}¥¥[¥¥]¥
¥|:;'<>¥¥.¥¥?¥¥/`]*$",
            "default":""
          },
          "ldevSetting":{
            "type":"object",
            "properties":{
              "fullAllocation":{
                "type":"string",
                "enum":[
                  "Enable",
                  "Disable"
                ],
                "default":"Disable"
              }
            }
          },
          "lunSetting":{
            "type":"object",
            "properties":{
              "lunStartsFrom":{
                "type":"integer",
                "description":"Enter a value in hex format.",
                "default":0,
                "minimum":0,
                "maximum":2047
              }
            }
          }
        }
      }
    }
  }
}
```

GUI メタ情報

```
{
  "displayName":"Volume Settings",
  "properties":{
    "values":{
      "displayName":"Volume Settings",
      "showLabel": false,
      "permission":"locked",
      "visibility":"exec",
      "items":{
        "displayName":"Volume Setting",
        "properties":{
          "usage":{
            "displayName":"Volume Usage",
            "permission":"unlocked",
            "visibility":"exec",
            "required": true
          },
          "numberOfVolumes":{
            "displayName":"Number of Volumes",
            "permission":"unlocked",
            "visibility":"exec",
            "required": true
          },
          "capacity":{
            "displayName":"Volume Capacity",
            "permission":"unlocked",
            "visibility":"exec",
            "required": true,
            "presentation":"capacityInGB"
          },
          "storageProfile":{
            "displayName":"Storage Profile",
            "permission":"unlocked",
            "visibility":"exec",
            "required": true,
            "presentation":"select"
          },
          "ldevLabel":{
            "displayName":"Volume Label",
            "permission":"unlocked",
            "visibility":"exec",
            "required": false
          },
          "ldevSetting":{
            "displayName":"LDEV Setting",
            "permission":"unlocked",
            "visibility":"exec",
            "properties":{
              "fullAllocation":{
                "displayName":"Full Allocation",
                "visibility":"exec",
                "permission":"unlocked",
                "required": true,
                "presentation":"select"
              }
            }
          },
          "lunSetting":{
            "displayName":"LUN Setting",
            "permission":"unlocked",
            "visibility":"exec",
            "properties":{
              "lunStartsFrom":{
                "displayName":"LUN Starts From",
                "visibility":"exec",
                "permission":"unlocked",
                "required": true
              }
            }
          }
```

```
            }
        }
    }
  }
}
```

4.5.7 出力プロパティを追加する

サービステンプレートに関連付けられたプロパティグループに出力プロパティを指定できます。

プロパティの一覧が示されます。

前提条件

開発状態のサービステンプレートが存在し、[フロー] ビューにステップが追加されている必要があります。

操作手順

1. [Service Builder Edit] 画面の [プロパティ] タブで、[追加] メニューから [出力プロパティ] を選択します。
   [サービスの出力プロパティ作成] ダイアログが表示されます。
2. [フロー] タブの [ステッププロパティ] エリアにある [値] フィールドの [鉛筆] アイコンをクリックし、[出力プロパティマッピング] ダイアログで出力プロパティを追加することもできます。
3. [出力プロパティ追加] ボタンをクリックします。
   [サービスの出力プロパティ作成] ダイアログが表示されます。
4. 出力プロパティに関連する詳細を入力します。
5. [OK] をクリックして、指定した出力プロパティの詳細を保存します。
   ここで指定した出力プロパティは、後で出力プロパティ一覧に反映されます。

関連参照

- 4.5.8 [サービスの出力プロパティ作成] または [サービスの出力プロパティ編集] ダイアログ

4.5.8 [サービスの出力プロパティ作成] または [サービスの出力プロパティ編集] ダイアログ

サービステンプレート用の出力プロパティは、[サービスの出力プロパティ作成] または [サービスの出力プロパティ編集] ダイアログから追加できます。

次の表は、[サービスの出力プロパティ作成] または [サービスの出力プロパティ編集] ダイアログのフィールド、サブフィールド、フィールドグループについての説明です。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 19 プロパティ定義フィールドグループ

フィールド	サブフィールド	説明
[プロパティキー:] *	-	出力プロパティキー。
[プロパティ名:] *	-	出力プロパティの名前。
[説明:]	-	出力プロパティの説明。
[プロパティグループ:]	-	プロパティが所属するプロパティグループを選択します。

フィールド	サブフィールド	説明
		[新しいプロパティグループを作成] を選択して、新しいプロパティグループを作成することもできます。
[表示/非表示:]	-	出力プロパティの表示設定を指定します。選択肢は次のとおりです。 • 表示 • 非表示
[データ型:]	-	プロパティのデータ型を選択します（string、boolean、integer、double、date、password、composite）。どのオプションを選ぶかに応じて、さまざまなオプションが新たに提示されます。 配列を扱う場合、[配列データ] オプションを選択して、データ型を配列として扱うことができます。こうすると、同じタイプのプロパティのセット（要素の数は変動あり）をひとつのプロパティとして扱うことができ、データマッピングが容易になります。特に、サービスと部品の間でデータのやりとりをする場合に容易になります。
[データ形式:]	-	データ形式を選択します。 • application/json • application/javascript • application/xml • text/html • text/plain • text/csv • application/octet-stream
[ドメインタイプ:]	-	リストからドメインタイプを選択します。または、[新規ドメインタイプ作成] をクリックし、[Create Domain Type Definition] ダイアログから関連する詳細を入力して、新しいドメインタイプを追加します。 このオプションは、データ型に [composite] を選択し、データ形式に [application/json] を選択すると使用できます。

アスタリスク（*）付きのフィールドは必ず指定します。

表 20 値と表現形式フィールドグループ

フィールド	サブフィールド	説明
[表現形式:]	-	プロパティの表示を指定します。表現形式は、データ型に応じてリストに表示されます。
[デフォルト値:]	-	プロパティのデフォルト値を指定します。指定できる値は、データ型によって異なります。
[表示条件:]	-	指定した条件を満たす場合、プロパティを表示します。
[活性化条件:]	-	指定した条件を満たす場合、プロパティを活性化します。

関連タスク

- 4.5.7 出力プロパティを追加する

4.5.9 変数を追加する

サービステンプレートと関連する特定のプロパティグループに変数を追加できます。

［プロパティグループ］の現在のプロパティ一覧は、［プロパティ］タブから表示されます。

前提条件

開発状態のサービステンプレートが存在し、［フロー］ビューにステップが追加されている必要があります。

操作手順

1. ［Service Builder Edit］画面の［プロパティ］タブで、［追加］プルダウンメニューから［変数］を選択します。
   ［変数作成］ダイアログが表示されます。
2. 追加する変数の詳細を入力します。［プロパティグループ］プルダウンメニューから、サービス共有プロパティを特定のプロパティグループに追加することもできます。データタイプおよび関連したデフォルト値を指定することもできます。
3. ［OK］をクリックします。
   指定した変数がサービステンプレートに追加されます。

次の作業

この変数を適切な出力値にマッピングすることを確認します。

関連参照

- 4.5.10 ［変数作成］または［変数編集］ダイアログ

4.5.10 ［変数作成］または［変数編集］ダイアログ

［変数作成］または［変数編集］ダイアログからサービステンプレート用の変数プロパティを入力または編集できます。

次の表は、［変数作成］または［変数編集］ダイアログのフィールド、サブフィールド、フィールドグループの説明です。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 21 プロパティ定義フィールドグループ

フィールド	サブフィールド	説明
［プロパティキー:］*	-	変数のキー名を指定します。
［プロパティ名:］*	-	変数名。
［説明:］	-	変数の説明。
［プロパティグループ:］	-	「デフォルトプロパティ」が設定されています。変数の場合、この項目を変更することはできません。
［データ型:］	-	プロパティのデータ型を選択します（string、boolean、integer、double、date、password、composite）。どのオプションを選ぶかに応じて、さまざまなオプションが新たに提示されます。例えば、date オプションを選ぶと、カレンダーインタフェースが提供されます。

フィールド	サブフィールド	説明
		配列を扱う場合、[配列データ] オプションを選択して、データ型を配列として扱うことができます。こうすると、同じタイプのプロパティのセット（要素の数は変動あり）をひとつのプロパティとして扱うことができ、データマッピングが容易になります。特に、サービスと部品の間でデータのやりとりをする場合に容易になります。
[データ形式:]	-	データ形式を選択します。 • application/json • application/javascript • application/xml • text/html • text/plain • text/csv • application/octet-stream
[ドメインタイプ:]	-	リストからドメインタイプを選択します。または、[新規ドメインタイプ作成] をクリックし、[Create Domain Type Definition] ダイアログから詳細を入力して、新しいドメインタイプを追加します。このオプションは、データ型に composite を選択し、データ形式に application/json を選択すると使用できます。

アスタリスク（*）が付いたフィールドは必須です。

表 22 値と表現形式フィールドグループ

フィールド	サブフィールド	説明
[デフォルト値:]	-	変数のデフォルト値を入力します。

関連タスク

- 4.5.9 変数を追加する

4.6 新しいサービステンプレートの作成例

このセクションでは、サービステンプレートのカスタマイズプロセスについて説明します。このサービステンプレートでは、特定のプラットフォームのボリュームをプロビジョニングするほか、ボリュームの割り当てが正常に終了したかどうかを示すメール通知を生成するコンポーネントのステップを追加します。

この例では、以下の手順を完了します。

1. 既存のサービステンプレートをコピーして、新しいサービステンプレートの詳細を入力します。
2. メール通知部品を追加して、環境に合わせて変更します。
3. コンポーネントのステップのフローを作成します。
4. 新しいサービステンプレートを作成します。

メモ この例では、service administrator がシステムのアーキテクチャをすでに考慮しており、目的のストレージサイズ、構成、I/O プロファイルに基づいてサービスの作成に必要な計算を実行しているものと仮定しま

す。テンプレートの値はベストプラクティスに基づくものですが、ユーザーの設定値は、ユーザーの特定のニーズに応じて異なる場合があります。

この例を開始するには、最初に［ツール］プルダウンメニューから［サービスビルダー］を選択して、［Service Builder Home］画面を表示する必要があります。［Service Builder Home］画面では、サービステンプレートと、それらに関連付けられている部品の作成と管理を行うための Service Builder オプションをすべて使用できます。

関連概念

- 4.1 サービステンプレート作成ワークフロー

関連タスク

- 4.2 新しいサービステンプレートを作成する
- 4.4.1 データフローでステップを作成する
- 4.5.4 入力プロパティを追加する
- 4.5.7 出力プロパティを追加する

4.6.1 既存のサービステンプレートのコピーを作成する

以下の手順は、Develop ロールを持つ service administrator が実行します。

操作手順

1. ［Service Builder Home］画面の［リリース］タブで、［カード］ビューまたは［テーブル］ビューからコピーするテンプレートを選択してハイライト表示します。この例では、「Allocate Volumes with Smart Provisioning」というサービステンプレートを選択してください。
2. ［複製］ボタンをクリックして、［サービステンプレート複製］ダイアログにアクセスします。
3. サービステンプレートの基本情報を次の表に示すように入力します。

表 23 ［サービステンプレート複製］ダイアログ

パラメーター	説明	値
［サービステンプレート ID:*］	-	コピーしたサービステンプレートのキー名を指定します。この例では、「NewTemplate」などの一意 ID を入力します。
［サービステンプレートバージョン*］	-	サービステンプレートのバージョン。コピーしたサービステンプレートの値がすでに入力されています。
［ ベンダー ID:*］	-	ベンダーを識別する何らかの名前を入力します。
［サービステンプレート名:*］	-	サービステンプレートのコピーバージョンに割り当てられた名前。コピーしたサービステンプレートの値がすでに入力されています。 この例では、「NewTemplate」などの名前を入力します。
［ベンダー名 : ］	-	コピーしたサービステンプレートのベンダー名（該当する場合）。
［説明 : ］	-	コピーしたサービステンプレートの説明。
［タグ : ］	-	サービステンプレートに関連付けられたタグカテゴリ。コピーしたサービステンプレートには、「Add New Storage」タグおよび「Configuration Manager」タグがすでに選択されています。

アスタリスク（*）が付いたフィールドは必須です。

4. ［OK］をクリックすると、変更された詳細を使用して新しいサービステンプレートが作成されます。コピーしたテンプレートに関連付けられたコンポーネント（部品およびほかのサービステンプレート）は、［Service Builder Edit］画面の［フロー］タブから表示されます。
既存の部品がコピー元のサービステンプレートからフローにステップとしてすでに追加されています。このステップをクリックすると、このステップに関連付けられた入出力プロパティを確認できます。

4.6.2 サービステンプレートのメール通知を追加する

サービステンプレートをコピーした後で、［Service Builder Edit］画面で必要な変更を加えることができます。

以下の手順に従って、メール通知部品を追加します。

操作手順

1. コンポーネント一覧からメール通知部品を見つけ、右の画面のフローエリアへドラッグします。必要に応じて、検索ボックスを使用して部品を見つけることもできます。［ステップ作成］ダイアログが表示されます。
2. ［ステップ作成］ダイアログで、部品ステップの適切な詳細情報を、次の表に示すように入力します。

表 24 ［ステップ作成］ダイアログ

パラメーター	説明	値
［ステップ ID：*］	-	メール通知部品の ID を指定します。この値はすでに入力されています。
［ステップ名:*］	-	部品ステップに割り当てられた名前。部品の値がすでに入力されています。
［説明：］	-	コピーしたサービステンプレートの説明。必要に応じてここに説明を入力できます。

アスタリスク（*）が付いたフィールドは必須です。

3. ［OK］をクリックして、選択した部品を追加します。画面のフローエリアに、新しく追加した部品ステップの画像が表示されます。
4. 新しく追加した部品ステップの実行順序を指定します。これを行うには、既存の部品の画像の横にある点をクリックし、矢印を［メール通知部品］ステップ上にドラッグします。矢印は、サービスの実行時にサービステンプレートが部品ステップをどの方向に向けて処理するかを示します。
5. 新しく追加した部品ステップの入力プロパティを指定します。例えば、「メール件名」プロパティに件名を入力したり、メッセージ本文に内容を追加したりできます。メール通知の値はユーザーごとに異なるため、「TO メールアドレス」プロパティの［サービスプロパティ］チェックボックスをオンにすることをお勧めします。これにより、ユーザーはサービスの実行時に［サービス編集］または［サービス実行］画面で、関連するメールアドレスを入力できます。

   ［サービス編集］画面と［サービス実行］画面の両方から関連するメールアドレスを入力する場合は、［プロパティ］タブで、「TO メールアドレス」プロパティの可視性を「［サービス作成］と［サービス実行］」に変更します。

4.6.3 新しいサービステンプレートをデバッグ、ビルド、テスト、およびリリースする

必要な部品の作成と追加が完了したら、サービステンプレートをビルドし、すべてのバグを解決した後、サービステンプレートをリリースできます。

操作手順

1. ［デバッグ］タブをクリックし、［OK］をクリックして、追加した部品を使用してサービステンプレートをビルドします。
   ［ビルド/リリース結果］ダイアログが表示され、サービステンプレートのビルド中に発生したエラーがすべて表示されます。
2. ビルドが正しく完了するまでトラブルシューティングを続けます。完了したら［閉じる］をクリックして、［ビルド/リリース結果］ダイアログを終了します。
   ビルドが成功すると、［デバッグ実行］ダイアログが表示されます。
3. サービステンプレートが正しく機能していることを確認します。これを行うには、リクエストを作成したりサービステンプレートのサービスリクエストを送信したりできるメインの Ops Center Automator ユーザーインタフェースに戻る必要があります。
4. ［サービス］タブで［作成］をクリックし、［サービステンプレート選択］画面で［すべてのバージョンを表示］をクリックして「NewTemplate」を表示させ、「NewTemplate」を選択し、［サービス作成］をクリックします。［サービス作成］画面で［保存して閉じる］をクリックします。
5. ［サービス］タブで、デバッグ状態の「NewTemplate」を選択し、［実行］をクリックします。
6. ［サービス実行］画面の［設定］ペインで、次の情報を入力します。

パラメーター	説明	値
［Volume Settings］		
［Configuration Manager Connection］	Configuration Manager の接続	Configuration Manager の接続を表から選択します。
［Host Settings］		
［Number of Hosts］	ボリュームを割り当てるホストの数	Single
［Host Name］	ホスト名	ホスト名を入力します。
［WWN Settings］	WWN 設定	加算記号（+）のアイコンをクリックして、情報を入力します。
［タスク設定］		
［タスク名］	タスクの名前	NewTemplate
［説明］	タスクの短い説明	サービス用のメールを生成するためのタスク。
［スケジュール種別］	タスクを実行するタイミング	即時実行

7. 必要な値をすべて指定したら、［実行］をクリックして、［サービス実行］の確認ダイアログで、［OK］をクリックします。
8. ［タスク］タブの［デバッグ］ビューで、「NewTemplate」タスクを選択し、［タスク詳細表示］をクリックして、タスクの概要、詳細、結果、ログ、ノートを参照します。
9. サービステンプレートをテストして、新しいサービステンプレートが正しく動作していることを確認してから、［Service Builder Edit］画面に戻り、［リリース］タブをクリックして、ユーザーがテンプレートをサブミットできるようにします。

5

新しい部品を作成する

既存の部品のコピーを作成してからニーズに合わせて変更するか、一から新しい部品を作成してコマンドを実行したり、スクリプトを実行したり、サービステンプレートの基本的なタスクを実行したりできます。

それぞれの部品には特定の目的があり、サービステンプレートでの部品の使用および順序は、サービステンプレートを作成する上で重要になります。

ここでは、部品を作成し編集する方法について説明します。

- 5.15 メールを生成する

5.1 部品作成のワークフロー

［Service Builder Home］画面の［部品の操作］プルダウンメニューから、新しい部品を管理、作成できます。

［部品の操作］プルダウンメニューから適切なオプションを選択することにより、既存の部品を編集、コピー、または削除できます。

新しい部品を作成するには、次のワークフローで説明しているさまざまなフェーズを経る必要があります。

フェーズ 1：準備

1. 部品の必要性と目的を決定します。プロセスを自動化するのに必要な機能ステップを検討し、1つまたは複数の部品が必要であるか、新規部品の作成と既存部品の修正のどちらが必要であるかを決定します。
2. 部品作成の準備をします。これには、部品の定義、アイコンファイル、そのタスクを実行するためのコマンドまたはスクリプトの準備、およびリソースファイルの準備が含まれます。

フェーズ 2：作成

1. ［Service Builder Home］画面で、新しい部品を作成するか、既存の部品をコピーして変更します。部品は開発状態から開始されます。
2. 基本情報を入力し、入力および出力プロパティを設定します。
3. 必要であれば、リモートコマンドと環境変数を設定します。

フェーズ 3：テスト

1. サービステンプレートの作成中に部品をフローに配置します。
2. サービステンプレートをテスト用にビルドします。
3. テストを完了します。
4. 部品をデバッグします。
5. 部品がサービステンプレート内で正しく実行されるまで、サービステンプレートの再ビルドと再テストを行います。

フェーズ 4：リリース

テストが完了したらサービステンプレートをリリースします。サービステンプレートと、関連する部品の状態はリリースに変更されます。

関連タスク

- 5.2 部品を作成する

5.2 部品を作成する

既存の（事前に準備された）部品の 1 つに基づいて部品を作成するか、サービステンプレートの特定のステップに対してコマンドまたはスクリプトを実行する新しい部品を、一から作成できます。

メモ　同じ部品キー名、ベンダー ID、およびバージョン番号を持つ部品は複製できません。

前提条件

- スクリプトの入力プロパティの処理方法、およびスクリプトから出力プロパティを引き渡す方法を決定します。
- スクリプトをファイル渡しで実行するのなら、スクリプトファイルをすぐに使えるように準備しておきます。

操作手順

1. ［Service Builder Home］画面の［部品の操作］プルダウンメニューから、［複製］を選択して既存の部品の 1 つをモデルとして使用するか、［作成］を選択して一から新しい部品を作成します。
   選択した作成方法に応じて、［部品複製］または［部品作成］ダイアログが開きます。
2. ［定義情報］タブで、部品の基本情報を入力します。
3. ［プロパティ］タブで、入力プロパティまたは出力プロパティのアイコンをクリックし、部品に関連付ける入力プロパティと出力プロパティを追加します。複数の入力プロパティを追加して、セクション内で入力プロパティをドラッグアンドドロップし、順序を変更できます。
4. ［リモートコマンド］タブで、［プラットフォームを追加］ボタンをクリックして、プラットフォームを選択します。そのあと、認証種別と部品に関連するその他の詳細を指定できます。
5. リモートコマンドやスクリプトを実行するために環境変数が必要な場合は、［詳細設定］をクリックしてから［追加］をクリックし、環境変数の名前（必須）と値を入力します。
6. 編集が完了したら、［保存］をクリックします。

操作結果

新しい部品が作成されて開発状態になり、［Service Builder Edit］画面の［フロー］タブからアクセスできるようになります。

関連概念

- 7.4 バージョンを管理する

関連タスク

- 5.9 部品のリモートコマンドを設定する

関連参照

- 5.3 ［部品作成］または［部品編集］ダイアログ

5.3 ［部品作成］または［部品編集］ダイアログ

新しい部品を作成するときは、［部品作成］または［部品編集］ダイアログの［定義情報］、［プロパティ］、または［リモートコマンド］タブから、関連する詳細を指定します。

［定義情報］タブ

［定義情報］タブで、現在選択されているサービステンプレートに関する一般的な詳細情報を参照、編集できます。

次の表で、［定義情報］タブから使用できるオプションについて説明します。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 25 ［部品作成］または［部品編集］の［定義情報］タブ

フィールド	サブフィールド	説明
［部品 ID］*	-	部品のキー名（最大 64 文字）。部品 ID とベンダー ID の組み合わせが 115 文字を超えることはできません。
［部品バージョン］*	-	部品のバージョン番号。
［ベンダー ID ］*	-	部品のベンダー ID（最大 64 文字）。部品 ID とベンダー ID の組み合わせが 115 文字を超えることはできません。
［部品名］	-	部品の名前（最大 64 文字）。
［ベンダー名］	-	部品のベンダー名（最大 64 文字）。
［説明］	-	部品の簡単な説明（最大 1,024 文字）。
［タグ］	-	部品に対して割り当てられたタグカテゴリ。部品のタグカテゴリを追加するには、加算記号（+）クリックします。
［アイコン］	-	部品に関連付けるアイコン画像（48 x 48 ピクセルの PNG ファイル）を指定します。 デフォルトの画像が表示されますが、［変更］ボタンをクリックして適切なファイル名を指定することで変更できます。［標準に戻す］ボタンを選択すると、いつでも元のアイコンに戻すことができます。

アスタリスク（*）が付いたフィールドは必須です。

［プロパティ］タブ

［プロパティ］タブで、サービステンプレートに関連付けられている入力プロパティと出力プロパティを検索、参照できます。プロパティのテキスト検索ボックスを使用して、プロパティを検索できます。入力ボタンまたは出力ボタンをクリックし（検索ボックスの隣）、入力プロパティと出力プロパティの一覧を切り替えます。編集の鉛筆アイコンをクリックすることにより、選択したプロパティを編集することもできます。

次の表で、［プロパティ］タブから使用できるオプションについて説明します。

表 26 ［部品作成］または［部品編集］の［プロパティ］タブ

フィールド	サブフィールド	説明
［プロパティキー］	-	プロパティに関連付けられたキーを表示します。
［プロパティ名］	-	プロパティに関連付けられた名前を表示します。
［説明］	-	プロパティについての説明を提供します。
［必須区分］	-	プロパティが必須（true）か必須ではない（false）かを示します。
［デフォルト値］	-	必要に応じてデフォルト値を指定します。

［リモートコマンド］タブ

［リモートコマンド］タブで、サービステンプレートの処理中に実行されるリモートコマンドを設定できます。

リモートコマンドの追加を開始するには、[プラットフォームを追加] をクリックし、選択した運用環境に関連する詳細を指定します。

次の表で、[リモートコマンド] タブから使用できるオプションについて説明します。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 27 [部品作成] または [部品編集] の [リモートコマンド] タブ

フィールド	サブフィールド	説明
[認証種別]	-	部品で必要とされる認証情報タイプを指定します。 サービスを実行するとき、[管理] タブの [エージェントレス接続先定義] ビューにある認証情報を使用する場合は、[エージェントレス接続先設定を使用]を選択します。認証情報タイプのデフォルト値は [エージェントレス接続先設定を使用] です。 [エージェントレス接続先設定を使用] の認証情報タイプでは、次の予約済みの部品プロパティが自動的に設定されます。 • plugin.destinationHost IPv4 アドレス、IPv6 アドレス、またはホスト名 (最大 256 文字) を使用して操作の対象を入力します。 [プロパティで指定] を選択して、認証情報を入力プロパティとして使用します。 次の予約部品プロパティは、[プロパティで指定] の入力プロパティの認証情報タイプに自動的に設定されています。 • plugin.destinationHost IPv4 アドレス、IPv6 アドレス、またはホスト名 (最大 256 文字) を使用して操作の対象を入力します。宛先のホストが Ops Center Automator サーバ (localhost) ならば、ユーザー ID とパスワードは必要ありません。 • plugin.account 対象ホストにログインするためのユーザー ID を入力します (最大 256 文字)。 • plugin.password 対象ホストにログインするためのパスワードを入力します (最大 256 文字)。 • plugin.suPassword Linux 環境で対象ホストにログインするために使用する root アカウントのパスワードを入力します (最大 245 文字)。対象ホストで Windows が実行されている場合、このプロパティは無視されます。
[Windows 接続用設定]	[システムアカウントで実行]	システムアカウントを使用してコマンドを実行します。
[Linux/UNIX 接続用設定]	[文字セット自動判定]	[文字セット自動判定] は Linux オペレーティングシステムに適用されます。有効にすると、ユーザーのデフォルトのロケールを使用してスクリプトが実行されます。無効にすると、LC_ALL=C のロケールでスクリプトが実行されます。デフォルト値は「有効」です。
	[root 権限で実行]	root 権限を使用して実行します。
[プラットフォームを追加]	-	リモートコマンドのプラットフォーム (Windows または Linux) を指定します。

フィールド	サブフィールド	説明
［定義情報の取り込み］	-	別のオペレーティングシステムから取得する必要のある設定を指定します。
［実行モード:］	-	スクリプトに基づく場合は［スクリプト］を、保存したコマンドを使用する場合は［コマンド］を選択します。
［コマンドライン:］*	-	リモートコマンドのコマンド（8,192 文字まで）を入力します。［実行モード］で［スクリプト］または［コマンド］を選択した場合は必須です。［入力プロパティを挿入］ボタンは、その部品について定義されているすべての入力プロパティをコマンドに挿入します。
［スクリプト設定:］	-	スクリプトファイルをアップロードする場合は［添付］を選択し、スクリプト情報を直接入力するには［直接入力］を選択します。スクリプトを実行する部品では、次のファイル形式を使用します。*<部品名>.<拡張子>*.
［ファイル］*	-	［選択］ボタンをクリックして、コマンドまたはスクリプトが格納されているファイルを指定します。
［出力プロパティのマッピング定義一覧］	-	コマンドまたはスクリプトの出力プロパティのマッピング定義を指定します。一覧からプロパティを選択してハイライト表示し、［編集］をクリックすることで、［出力フィルタ編集］ダイアログにアクセスでき、出力プロパティに渡すデータを制御する出力フィルタを指定できます。
［実行ディレクトリ］	-	実行フォルダを指定します。
［環境変数］	-	コマンドまたはスクリプトの環境変数を選択します。このオプションを選択すると［環境変数作成］または［環境変数編集］ダイアログが表示され、適切な変数情報を入力できます。

アスタリスク（*）が付いたフィールドは必須です。

関連概念

- 7.4 バージョンを管理する

関連タスク

- 5.2 部品を作成する

関連参照

- 付録 A.4 部品用のロケール設定

5.4 部品のプロパティについて

部品用に入力と出力のプロパティを定義して、タスクの実行中に必要なパラメーターを指定し、結果を処理します。

すべてのステップのコンポーネントの入力と出力プロパティを、サービステンプレートの入力と出力プロパティおよび変数にマッピングする必要があります。サービステンプレートで使用されるコンポーネントの入力プロパティは、入力プロパティまたは変数、もしくはサービステンプレートに関連付けられた、その他のコンポーネントのステップの出力プロパティと関連付ける必要があります。サービステンプレートの入力プロパティはサービスの実行に必要な入力値を格納します。サービステンプレートの出力プロパティはサービスの実行結果を格納します。

部品の入力プロパティの値は、そのプロパティにこれまでに設定されている値、もしくはサービステンプレート用に直接設定した値です。入力プロパティは変数にすることもできます。サービス共有プロパティを入力プロパティに適用することもできます。

出力プロパティの内容はコンポーネントのタイプによって異なります。ステップ実行の結果を出力プロパティとして格納することができます。変数は、コンポーネント間で受け渡しされる値を一時的に保持します。

コンポーネントの入力プロパティと出力プロパティは、[Service Builder Edit]画面の[フロー]タブで設定されます。サービステンプレートの入力プロパティと出力プロパティおよび変数は、[Service Builder Edit]画面の[プロパティ]タブで設定されます。

入力と出力プロパティは、データタイプとして composite を指定しないかぎりは最大で 1,024 文字まで格納できます。composite データタイプを指定せずに 1,024 を超える値を指定した場合には、最初の 1,024 文字がプロパティ値として格納され、残りは捨てられます。プロパティキーの値を?dna_property-key?フォーマットで参照する場合には、参照された値が 1,024 文字より長い場合には切り捨てられます。

プロパティキーと目的があらかじめ決まっている部品プロパティは予約済み部品プロパティと呼ばれます。このようなプロパティはリモートコマンドの認証情報および実行対象ホストを指定します。

関連参照

- 4.4.3 ステッププロパティを指定する

5.5 部品の入力プロパティを追加する

入力プロパティには、リモートコマンドの引数や操作の対象ホストのような、部品が実行中に必要とする値を格納します。入力プロパティは、[部品作成]ダイアログまたは[部品編集]ダイアログで作成され、定義されます。

[Service Builder Home]画面で、[部品の操作]プルダウンメニューから[作成]オプションまたは[編集]オプションを選択します。

[部品編集]ダイアログの[プロパティ]タブで、(入力プロパティアイコンをクリックして)入力リストが選択されていることを確認し、[追加]ボタンをクリックして必要な入力プロパティを入力します。複数の入力プロパティを追加して、セクション内で入力プロパティをドラッグアンドドロップして順序を変更できます。

続いて部品のプロパティとリモートコマンドを入力します。

関連参照

- 5.6 [部品の入力プロパティ作成]または[部品の入力プロパティ編集]ダイアログ

5.6 [部品の入力プロパティ作成]または[部品の入力プロパティ編集]ダイアログ

次の表で、[部品の入力プロパティ作成]または[部品の入力プロパティ編集]ダイアログのフィールド、サブフィールド、およびフィールドグループについて説明します。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 28 ［部品の入力プロパティ作成］または［部品の入力プロパティ編集］ダイアログ

フィールドグループ	フィールド	説明
［プロパティ定義］	［プロパティキー:］*	入力プロパティキーの名前。
	［プロパティ名:］*	入力プロパティの名前。
	［説明:］	入力プロパティの説明。
	［可視性:］	入力プロパティを［サービス作成］および［サービス実行］画面の両方に表示するのか（[サービス作成とサービス実行]）、［サービス作成］画面だけに表示するのか（[サービス作成]）を選びます。
	［表示設定:］	入力プロパティの表示設定を指定します。選択肢は次のとおりです。 • 変更可能 • 変更不可 • 非表示
	［必須区分:］	このチェックボックスが選択された場合、プロパティが必須であることを示します。
	［データ型:］	プロパティのデータ型を選択します（string、boolean、integer、double、date、password、composite）。どのオプションを選択するかによって、データ入力の際の制約条件を指定する様々なオプションが提示されます。 配列を扱う場合、［配列データ］オプションを選択して、データ型を配列として扱うことができます。こうすると、同じタイプのプロパティのセット（要素の数は変動あり）を 1 つのプロパティとして扱うことができ、データマッピングが容易になります。特に、サービスと部品の間でデータのやりとりをする場合に容易になります。 ［ファイル参照］チェックボックスを選択すると、プロパティの値をファイルパスとして表すように指定します。プロパティの値は自動的にファイルに格納され、ファイルのパスを入力プロパティとして取得することができます。例えば、コマンドラインで、直接値を入力する代わりにファイルパスを使用できます。
	［データ形式:］	データ形式を選択します。 • application/json • application/javascript • application/xml • text/html • text/plain • text/csv • application/octet-stream
	［ドメインタイプ:］	プルダウンリストからドメインタイプを選択します。あるいは、加算記号（+）をクリックし、［Create Domain Type Definition］ダイアログから関連情報を入力して、新しいドメインタイプを追加します。このオプションは、データ型に［composite］を選択し、データ形式に［application/json］を選択すると使用できます。

フィールドグループ	フィールド	説明
［制約条件］	［最小値/最小長:］	integer と double の最小値を指定します。データ型が string または password であれば、プロパティの最小の長さを入力します。データ型が［date］の場合、一番早い日付を入力します。
	［最大値/最大長:］	integer と double の最大値を指定します。データ型が string または password であれば、プロパティの最大の長さを入力します。データ型が［date］の場合、一番遅い日付を入力します。
	［入力文字制限:］	データ型が string または password であれば、正規表現を使用して、許可する文字を入力します。 例：^[0-9a-zA-Z¥.¥-]*$
	［最小配列長:］	配列の要素の最小長を指定します。
	［最大配列長:］	配列の要素の最大長を指定します。
	［バリデーションスクリプト:］	記述された JavaScript に基づき、プロパティを検証します。
［値と表現形式］	［表現形式:］	選択されたデータ型によって、どのようにプロパティの選択が提示されるかを決定するオプションを指定します。
	［デフォルト値:］	プロパティのデフォルト値を指定します。指定できる値は、データ型によって異なります。 ［配列データ］オプションを指定する場合、デフォルト値は、角括弧で囲まれたコンマ区切りの文字列値で入力する必要があります。 例：["1","2","3"]
	［データソース:］	データが［静的］であるか、［動的］かつ外部リソースプロバイダから取得されたものかどうかを指定します。
	［リスト表示値:］	プロパティのデータソースを静的に取得する場合（［データソース］に［静的］オプションを選択した場合）、リスト表示値を指定します。
	［外部リソース:］	プロパティのデータソースが動的に取得する場合（［データソース］に［動的］オプションを選択した場合）、外部リソースプロバイダを指定します。 プルダウンリストで、外部リソースプロバイダを追加、編集、アップロード、または削除することもできます。
	［追加パス:］	リクエスト URL の追加パス部分を指定します。不要な場合は、空のままにしておきます。追加パスは、次に示すように、URL 内で外部リソースプロバイダ ID の後に続くパスのことです。 /Automation/v1/objects/ExternalResources/<*外部リソースプロバイダ ID* >/<*追加パス*>?<*クエリパラメタ*>
	［クエリパラメタ:］	外部リソースプロバイダのためのクエリパラメタを指定します。`serviceID` プロパティと `serviceTemplateID` プロパティは自動的に追加されます。他のプロパティのプロパティ値を埋め込むために、{$ref:*keyName*}を指定することができます。JSON 値の場合、{$ref:*keyName#json path*}を指定できます。
	［名前フィールド:］	セレクトのラベルを使用するには、リソース列のフィールド名を指定します。省略した場合、名前フィールドが使用されます。
	［値フィールド:］	セレクトのラベルを使用するには、リソース列のフィールド名を指定します。省略した場合、instance ID フィールドが使用されます。

フィールドグループ	フィールド	説明
	[表示条件:]	指定した条件を満たす場合、プロパティを表示します。
	[活性化条件:]	指定した条件を満たす場合、プロパティを活性化します。

アスタリスク（*）が付いたフィールドは必須です。

関連概念

- 5.5 部品の入力プロパティを追加する

5.7 部品の出力プロパティを追加する

出力プロパティには、リモートコマンドの引数や操作の対象ホストのような、部品が実行中に必要とする値を格納します。出力プロパティは、[部品作成] ダイアログまたは [部品編集] ダイアログで作成され、定義されます。

[Service Builder Home] 画面で、[部品の操作] プルダウンメニューから [作成] オプションまたは [編集] オプションを選択します。

[部品編集] ダイアログの [プロパティ] タブで、(出力プロパティアイコンをクリックして) 出力リストが選択されていることを確認し、[追加] ボタンをクリックして必要な出力プロパティを入力します。複数の出力プロパティを追加して、セクション内で出力プロパティをドラッグアンドドロップして順序を変更できます。

続いて部品の入力プロパティと出力プロパティおよびリモートコマンドを入力します。

関連参照

- 5.8 [部品の出力プロパティ作成] または [部品の出力プロパティ編集] ダイアログ

5.8 [部品の出力プロパティ作成] または [部品の出力プロパティ編集] ダイアログ

サービステンプレートの出力プロパティを追加したり修正したりできます。

次の表で、[部品の出力プロパティ作成] または [部品の出力プロパティ編集] ダイアログのフィールド、サブフィールド、およびフィールドグループについて説明します。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 29 [部品の出力プロパティ作成] または [部品の出力プロパティ編集] ダイアログ

フィールドグループ	フィールド	説明
[プロパティ定義]	[プロパティキー:] *	出力プロパティキー。
	[プロパティ名:] *	出力プロパティの名前。
	[説明:]	出力プロパティの説明。
	[表示/非表示:]	出力プロパティの表示設定を指定します。選択肢は次のとおりです。

フィールドグループ	フィールド	説明
		• 表示 • 非表示
	[データ型:]	プロパティのデータ型を選択します（string、boolean、integer、double、date、password、composite）。どのオプションを選ぶかに応じて、さまざまなオプションが新たに提示されます。 配列を扱う場合、[配列データ] オプションを選択して、データ型を配列として扱うことができます。こうすると、同じタイプのプロパティのセット（要素の数は変動あり）を 1 つのプロパティとして扱うことができ、データマッピングが容易になります。特に、サービスと部品の間でデータのやりとりをする場合に容易になります。 [ファイル参照] チェックボックスを選択すると、プロパティの値をファイルパスとして表すように指定します。ファイルを経由して次のステップで出力値を使用できます。例えば、部品で次のコマンドを実行し、[ファイル参照] で「?dna_output?」がチェックされている場合、パスが「?dna_output?」のファイルに出力値を保存し、次のステップの入力値として使用できます。 **<command> > "?dna_output?"**
	[データ形式:]	データ形式を選択します。 • application/json • application/javascript • application/xml • text/html • text/plain • text/csv • application/octet-stream
	[ドメインタイプ:]	リストからドメインタイプを選択します。あるいは、加算記号（+）をクリックし、[Create Domain Type Definition] ダイアログから詳細を入力して、新しいドメインタイプを追加します。このオプションは、データ型に「composite」を選択し、データ形式に「application/json」を選択すると使用できます。
[値と表現形式]	[表現形式:]	選択されたデータ型によって、どのようにプロパティの選択が提示されるかを指定します。
	[表示条件:]	指定した条件を満たす場合、プロパティを表示します。
	[活性化条件:]	指定した条件を満たす場合、プロパティを活性化します。

アスタリスク（*）が付いたフィールドは必須です。

関連概念

- 5.7 部品の出力プロパティを追加する

5.9 部品のリモートコマンドを設定する

部品はリモートコマンドを使用して、入力プロパティをスクリプトまたはコマンドに渡します。また、リモートコマンドはデフォルトの結果出力の出力プロパティにフィルターをかけるのにも使用

されます。標準出力から希望する値を格納する出力フィルターを設定します。部品は 1 つ以上のリモートコマンドを必要とします。環境変数はリモートコマンドを通じて設定されます。

操作手順

1. ［Service Builder Home］画面で、［部品の操作］プルダウンメニューから［作成］オプションまたは［編集］オプションを選択します。既存の部品を編集している場合、カードビューまたはテーブルビューから部品を選択し、［編集］をクリックします。
［部品作成］ダイアログまたは［部品編集］ダイアログが表示されます。
2. ［リモートコマンド］タブで、［プラットフォームを追加］ボタンをクリックして、適切な運用環境を選択します。
オペレーティングプラットフォームの選択肢が提供されます。
3. リストから適切なプラットフォームを選択して、関連する詳細を入力します。
選択したプラットフォームに応じて選択肢が提供されます。
4. リモートコマンドに関連するすべての詳細を入力して、［保存］をクリックします。
選択した部品のリモートコマンドが作成されます。

5.10 環境変数を設定する

［部品作成］ダイアログまたは［部品編集］ダイアログの［リモートコマンド］タブから部品を作成または編集するときに、環境変数を設定できます。

操作手順

1. ［Service Builder Home］画面で、［部品の操作］プルダウンメニューから［作成］オプションまたは［編集］オプションを選択します。
新しい部品の詳細を入力、または［部品一覧］ダイアログから編集する既存の部品を選択すると、［部品作成］ダイアログまたは［部品編集］ダイアログが表示されます。
2. ［リモートコマンド］タブの［詳細設定］セクションで、［環境変数］ボックスの下の［追加］をクリックします。
［環境変数作成］ダイアログが表示されます。
3. 環境変数に関連する属性（変数名と値）を入力し、［OK］をクリックします。
定義した環境変数およびそれに関連付けられた値が表示されます。必要に応じて、環境変数の追加や既存の環境変数の編集を続けることができます。

関連参照

- 5.11 ［環境変数作成］または［環境変数編集］ダイアログ

5.11 ［環境変数作成］または［環境変数編集］ダイアログ

［部品作成］または［部品編集］ダイアログボックスからアクセスできる［リモートコマンド］タブの［詳細設定］セクションから、リモートコマンドの環境変数を入力または修正できます。

次の表は、［環境変数作成］または［環境変数編集］ダイアログのフィールド、サブフィールド、フィールドグループについての説明です。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 30 ［環境変数作成］または［環境変数編集］ダイアログ

フィールド	サブフィールド	説明
［変数名：］*	-	環境変数の名前。
［値：］	-	変数の値。

アスタリスク（*）が付いたフィールドは必須です。

関連タスク

- 5.10 環境変数を設定する

5.12 出力フィルターを追加する

標準出力および標準エラー出力は、出力プロパティに格納されます。したがって、リモートコマンドに正規表現を使用しておき、標準出力の結果にフィルターを設定して希望の値を得ることを推奨します。

例

デフォルトの出力プロパティからディスク ID を得るには、次のステップに従います。

1. 出力プロパティにフィルターを設定します。
   `diskid:n` はスクリプトの結果の出力メソッドです
   `diskid:(.+)` は出力プロパティにフィルターをかける正規表現です
2. スクリプトを作成して結果を渡します。
3. リモートコマンドを使って出力フィルターを設定しますが、ここで、
   `diskid:(.+)` は出力フィルターです
   `diskid:1` は標準出力の文字列です
   `blank` は出力プロパティの値です

スクリプトの実行結果：

- `diskid:1` は標準出力の文字列です
- *`1`* は出力プロパティの値です

［部品複製］または［部品編集］ダイアログからアクセスできる［リモートコマンド］タブで出力フィルタを指定します。

出力フィルターを設定するには：

［出力プロパティのマッピング定義一覧］ボックスで、出力プロパティの行を選択してハイライト表示し、編集（鉛筆）アイコンをクリックします。［出力フィルタ編集］ダイアログで、関連する詳細を入力します。

必要なら、続けて環境変数を設定します。

メモ　正規表現で複数グループを指定する場合には、最初のグループに一致する値だけが出力プロパティに格納されます。さらに、複数の値範囲に正規表現を適用する場合には、値の最初の範囲だけが出力プロパティに格納されます。

メモ　正規表現はマルチラインモードで評価されます。不要な文字（行送りなど）を除去するには、シングルラインモードを指定して正規表現を書く必要があります。シングルラインモードは、正規表現では「(?s)」として記述されます。
正規表現により抽出されるのは、括弧でグループ化された部分です。複数のグループが存在する場合、抽出対象は最初のグループになります。正規表現を使用して標準出力を抽出するいくつかの例を示します。
単一行からの文字列抽出例：

標準出力
```
server:sv001
CPU - 89%
Memory - 77%
```

正規表現
```
server:(.*)
```

抽出結果
```
sv001
```

複数行からの文字列抽出例 1：

標準出力
```
server:sv001
CPU - 89%
Memory - 77%
```

正規表現
```
server:(?s)(.*)
```

抽出結果
```
sv001
CPU - 89%
Memory - 77%
```

複数行からの文字列抽出例 2：

標準出力
```
server:sv001
CPU - 89%
Memory - 77%
```

正規表現
```
server:(?s)(.*)¥sMemory
```

抽出結果
```
sv001
CPU - 89%
```

関連参照

- 5.13 ［出力フィルタ編集］ダイアログ

5.13 ［出力フィルタ編集］ダイアログ

正規表現を使用して出力フィルタを指定し、出力プロパティによって処理されるデータを制御できます。出力フィルタが空の場合、標準出力と標準エラー出力が出力プロパティに直接格納されます。

次の表は、［出力フィルタ編集］ダイアログのフィールド、サブフィールド、およびフィールドグループについての説明です。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 31 ［出力フィルタ編集］ダイアログ

フィールド	サブフィールド	説明
［出力フィルタ：］	-	出力プロパティによって格納されるデータを絞り込むための正規表現を入力します。
［出力フィルタの動作確認］	-	処理を実行し、フィルタによるデータの処理が意図したように行われることを確認します。結果とともに、標準出力の詳細が表示されます。

関連タスク

- 5.12 出力フィルターを追加する

5.14 分岐部品を使用して条件分岐を作成する

サービステンプレート内のあるステップが特定の条件を満たす場合にのみ実行されるように、条件分岐を作成できます。

条件分岐は、以前のステップの処理中に発生する状態に基づいて適切なステップを実行するために役立ちます。

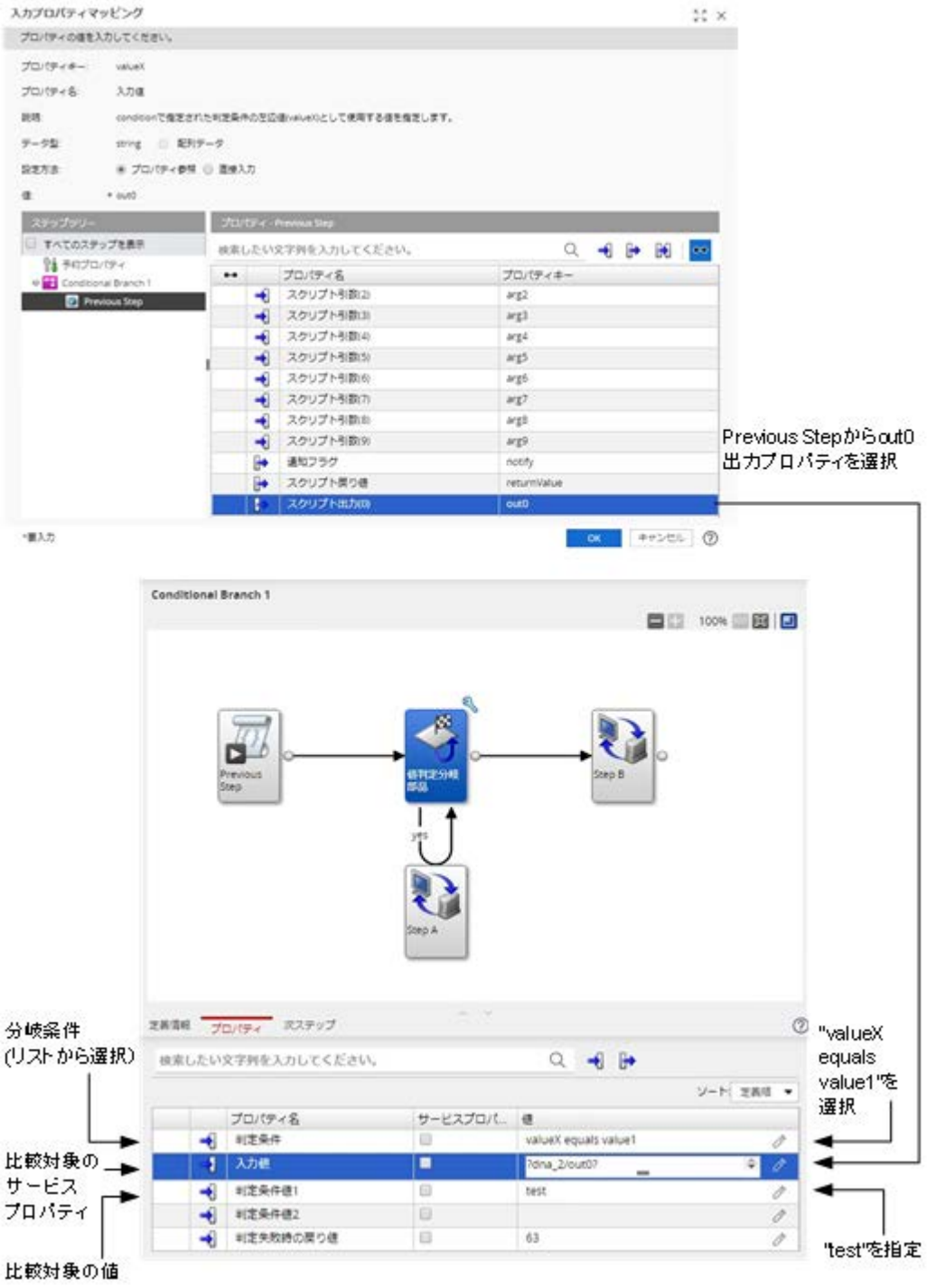
条件分岐を作成するために次の部品を提供しています。

- 値判定分岐部品：サービスプロパティと指定値とを比較し（必要な場合には、変数またはステッププロパティとも比較できます）、次にその結果に応じた動作をします。
- 戻り値判定分岐部品：以前のステップで生成された戻り値と指定値とを比較し、 次にその結果に応じた動作をします。

条件分岐を作成するには：

1. 分岐が起きるステップの後のフロー中に、適切な判断部品を挿入します。
2. 条件を満たす場合に実行すべきステップを分岐と続きのステップの場所との中間に置きます。
3. ステップの間を線で結び、実行のフローを定義します。
4. 入力値と出力値をセットして条件を定義し、適切な値を指定します。

次の図は、以前のステップが生成したステッププロパティが与えられた値と等しい場合の条件分岐を設定する値判定分岐部品を使う例を示しています。

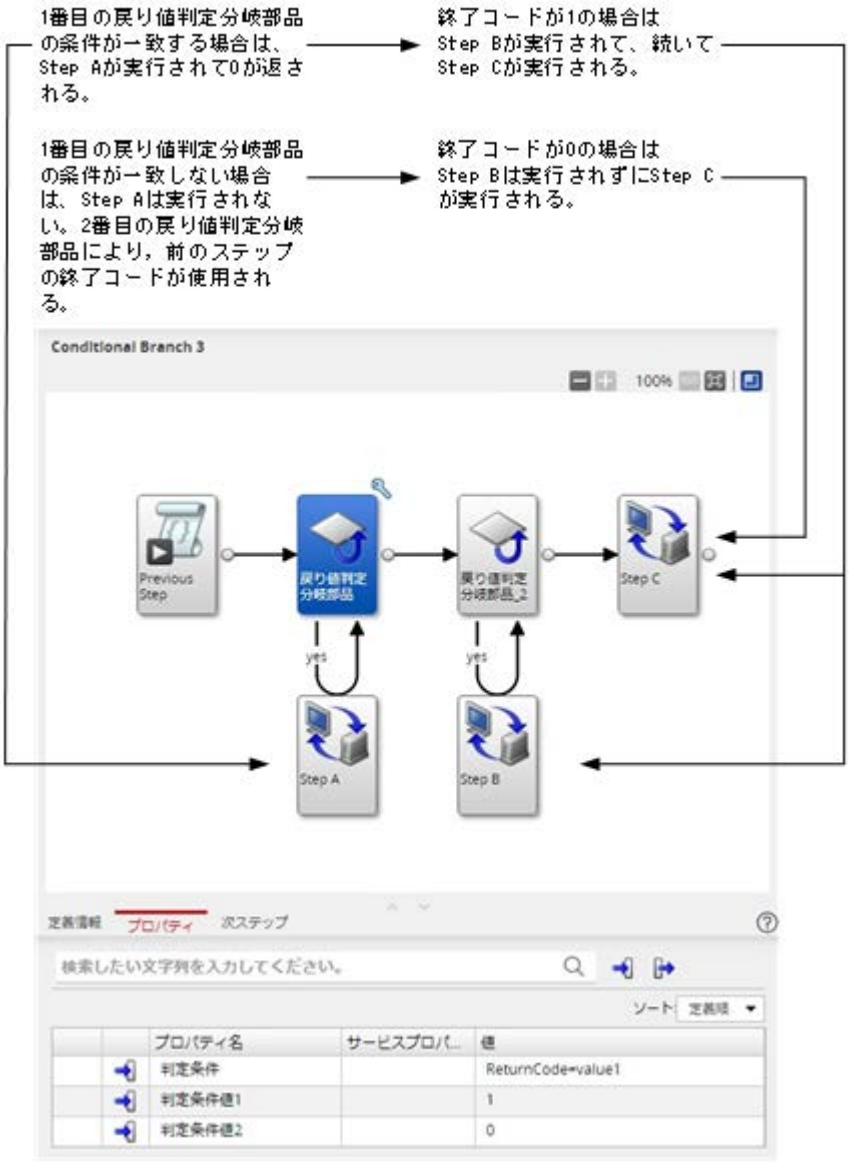


この例で「Step A」は、以前のステップからの「out0」が「test」に等しい場合にのみ実行されます。

次の図は、以前のステップが生成した終了コードが指定した基準を満たす場合の条件分岐を設定する戻り値判定分岐部品を使う例を示しています。

この例で「Step A」は、以前のステップからの終了コードが「0」に等しい場合にのみ実行されます。

さらに複雑な条件分岐を作成するには、複数の戻り値判定分岐部品を使用して、特定の条件が満たされる場合にあるステップを実行し、条件が満たされない場合に別のステップを実行させます。



5.15 メールを生成する

サービステンプレートの処理中に適切な部品を使用して、指定した通知または何らかの出力の内容を含めたメールを生成できます。

メール通知部品を使用して、特定の条件が発生したことを通知するメールや、タスクまたはプロセスから出力された情報を含めたメールを、指定した受信者に送信できます。例えば、Allocate Volume Service に関連付けられた LUN パス構成の結果を、指定したメールアドレスに送信できます。この部品を使用する場合、メールの受信者のアドレス、件名行、および本文を指定します。本文は、あらかじめ定義したメッセージまたは前のステップからの出力で構成することができます。メールの書式化のための文字コードを指定することもできます。

メールの送信者に関する詳細は、SMTP サーバの設定で定義します。メール通知部品を使用してメールを送信する前に、メール設定でメールの正しい詳細を指定する必要があります。この設定には、［管理］タブの［システム設定］からアクセスできます。

メールの生成方法

1. フロー内で、メール通知の情報が含まれるステップ以降の場所にメール通知部品ステップを挿入します。
2. ステップの間を線で結び、実行のフローを定義します。
3. メール通知部品ステップをクリックし、各入力プロパティに応じて適切な値を入力し、受信者のアドレス、文字コード、および件名を指定します。
4. メール通知部品の Body フィールドで、メールの内容を生成する部品ステップ内の出力プロパティのキーを入力します。

場合によって、デフォルトの JSON 形式ではない出力を送信する必要があり、その出力をメールの送信前に最初に変換しなければならないことがあります。これを行うには、以下のどちらかの部品を使用できます。

- JavaScript 実行部品：この部品は JSON 入力データをオブジェクトとして扱うことができるため、そのオブジェクトを、JavaScript スクリプトを使用して必要な形式に変換できます。
- File Adapter 部品：この部品は JSON 入力から値を取り出すことができます。

メール通知の生成方法に関する詳細情報については、新しい部品の作成例のトピックで提供されている段階的な指示を参照してください。

6

ビルド、デバッグ、およびリリース

新しい、または変更されたサービステンプレート用のステップを指定し、フローを作成した後で、[デバッグ] タブでビルドを生成する必要があります。ビルドが正常に完了すると、ビルトインのデバッガーを使ってステップを実行して、実行フローまたはプロパティマッピングに発生する問題を修正できます。サービステンプレートが正しく機能している場合には、運用環境に向けてリリースすることができ、サービスの作成に利用できます。

ここでは、サービステンプレートおよび部品をデバッグしリリースする方法について説明します。

6.1 デバッグとリリースのワークフロー

サービステンプレートを作成または修正すると、次のフェーズに沿ってサービスをビルドし、デバッグできます。

フェーズ 1 : 準備

サービステンプレートのビルドに先立って、サービステンプレートの詳細が指定されていることを確認します。すなわち、適切な部品ステップが実行順に整備されていること、および入力プロパティと出力プロパティの値が適切に定義されていることを確認します。

フェーズ 2 : ビルド

1. ［デバッグ］タブをクリックして、サービステンプレートのビルドを開始します。
2. ［ビルド/リリース結果］ダイアログから返される情報を確認して、エラーが発生しているかどうかを見ます。
3. もしもエラーがあれば、エラーメッセージ上にカーソルを持っていき、エラーに関する詳細を見ます。
4. 何か必要な修正があれば修正を行い、ビルドが正常終了となるまで、サービステンプレートの再ビルドを続けます。

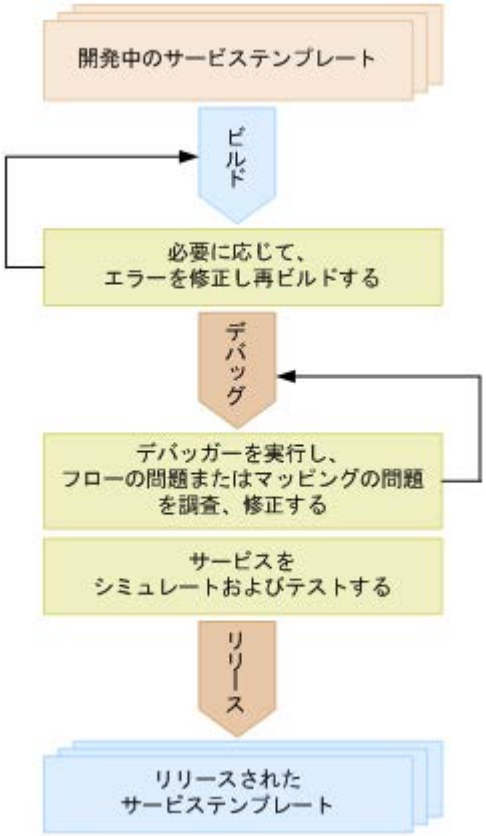
フェーズ 3 : デバッグ

1. サービステンプレートが正常にビルドされると、必要なデバッグを実行できます。
2. すべてのエラーが解決されるまで、エラーを修正してサービステンプレートの再ビルドを続けます。
3. すべての問題が解決した後に、開発環境にサービスを追加、実行してテストを実施します。

フェーズ 4 : リリース

1. サービステンプレートが適切に機能していて、さらに何も問題が発生しないのであれば、サービステンプレートをリリースできます。操作環境にサービステンプレートをサブミットするには、サービステンプレートがリリース状態である必要があります。
2. リリースされたサービステンプレートからサービスを作成します。
3. タスクの実行結果をチェックします。何か問題が見つかった場合には、影響があるサービステンプレートまたは部品を修正して、デバッグプロセスを繰り返します。

次の図に、サービステンプレートのデバッグ、テスト、リリースを行う際に従う一般的なステップを示します。



6.2 サービステンプレートをビルドする

サービステンプレートが作成され開発状態にあると、次のステップはサービステンプレートおよび関連する部品をビルドしてデバッグすることです。

サービステンプレートのビルドとデバッグを開始するためのステップを示します。

前提条件
開発状態にあるサービステンプレートが存在しなければなりません。

操作手順

1. ［Service Builder Edit］画面で、［デバッグ］タブをクリックします。
   デバッグ実行の確認画面を確認したあと、［ビルド/リリース結果］ダイアログが表示され、サービステンプレートのビルドプロセスが実行されます。エラーが生成された場合、エラーのアイコンとリンクが表示され、必須プロパティの入力に失敗したときには、警告アイコンが表示されます。
2. 指示が出ているところに必要な修正を行い、必要に応じて操作の調整を行います。
3. サービステンプレートにエラーが出なくなるまで、ビルドプロセスを繰り返します。

操作結果

サービステンプレートのビルドが終わると、次の処理が発生します。

- サービステンプレートのデバッグバージョンに追加したサービスは削除され、このサービスから実行されたタスクはアーカイブされます。
- デバッグバージョンのサービステンプレートから実行されたデバッグタスクは削除されます。

- インポートされたサービステンプレートのデバッグバージョンは削除されて、再インポートされます。

ビルドプロセスが正常終了すると、[デバッグ実行] ダイアログが表示され、実行のフローをチェックしたり、サービステンプレートのリリース前に必要な調整を加えたりできます。

次の作業

- デバッガーにアクセスしてステップのフローをチェックし、必要な修正を行います。
- サービステンプレートのデバッグ構成に基づいてサービスとタスクを作成します。サービスの作成に関する詳細情報については、『Hitachi Ops Center Automator ユーザーズガイド』を参照してください。
- サービステンプレートが正しく動作し、ビルドプロセスに合格したなら、サービステンプレートのリリースへと進みます。

関連タスク

- 6.14 サービステンプレートをリリースする

関連参照

- 6.3 [ビルド/リリース結果] ダイアログ

6.3 [ビルド/リリース結果] ダイアログ

サービステンプレートのビルドとリリースを行うとき、[ビルド/リリース結果] ダイアログで結果を最初にチェックします。

次の表で、[ビルド/リリース結果] ダイアログのフィールド、サブフィールド、およびフィールドグループについて説明します。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

表 32 [ビルド/リリース結果] ダイアログ

フィールド	サブフィールド	説明
[概要]	[ビルド/リリース結果]	サービステンプレートのビルド結果を表示します。
	[状態]	サービステンプレートの現在の状態を表示します。
[詳細]	[種別]	状態メッセージのタイプを示します。
	[メッセージ ID]	メッセージ ID を示します。 スクリプトを実行する部品では、次のファイル形式を使用します。 <部品名>.<拡張子>
	[メッセージ]	状態メッセージの内容を示します。

サービステンプレートのビルドが正常に終了してから、[デバッグ実行] ダイアログで、デバッグプロセスを開始できます。

関連タスク

- 6.2 サービステンプレートをビルドする
- 6.14 サービステンプレートをリリースする

6.4 デバッガーを実行する

サービステンプレートを正しくビルドした後で、［デバッグ実行］ダイアログの［OK］をクリックして、ビルトインのデバッガーを実行できます。デバッグインタフェースにアクセスする前に、［デバッグ実行］ダイアログでデバッグサービスとタスクの適切な詳細を指定する必要があります。また、［サービス作成］や［サービス実行］画面で、サービスのプロパティを確認および編集する、または変更を加えることができます。

ビルトインのデバッガーを使用することで、部品とサービステンプレートのフローが意図したとおりに機能するかどうかを確認できます。デバッグセッションでは、以下を行うことができます。

- サービステンプレートのフロー内のステップの実行を制御し、問題を個別化して修正することができます。
- デバッグタスクを実行および管理しながら、すべての階層レベルでフローの推移をチェックできます。
- プロパティのマッピングが正しく設定されており、後続のステップを実行するための条件が意図した通りに流れることを確認できます。
- 現在実行中のタスクの入力および出力プロパティを修正できます。
- ブレークポイントを設定し、フロー内で指定したステップから処理を開始したり、その前に処理を終了させたりできます。
- 特定の部品の処理（スクリプト/コマンド、繰り返し、ユーザーの応答待ち）をスキップし、部品の処理が実行されたかのように次のステップに進むことができます。これにより、次のステップの実行条件に従ってフローを継続させることができます。
- 繰り返し実行されるフローの結果を表示できます（各実行時に）。
- 実行中のタスクの実行時ログを参照できます。
- ［サービス作成］、［サービス編集］、および［サービス実行］画面の項目を編集し、サービステンプレートの処理をシミュレートできます。
- 部品に問題を検出した場合、任意のプロパティ値または戻り値を部品に割り当て、再実行できます。これにより、指定したプロパティ値または戻り値が部品処理やフロー推移に与える影響を見ることができます。

サービステンプレートのデバッグは何回でも実行できます。

サービステンプレートをデバッグすると、Ops Center Automator はデバッグサービスとデバッグタスクを作成します。

デバッグサービス

デバッグサービスは、サービステンプレートのデバッグ時に生成され実行されるサービスです。デバッグサービスはサービステンプレートごとに生成されます。すでにデバッグ処理を行ったサービステンプレートをデバッグすると、Ops Center Automator は既存のデバッグサービスを削除して新しいデバッグサービスを作成します。デバッグサービスは、［タスク］－［デバッグ］ビューの［サービス名］列には表示されますが、［サービス］画面には表示されないことに注意してください。

デバッグタスク

デバッグタスクは、サービステンプレートのデバッグ時にデバッグサービスに生成されるタスクです。すでにデバッグ処理を行ったサービステンプレートをデバッグすると、Ops Center Automator は既存のデバッグタスクを削除して新しいデバッグタスクを作成します。デバッグタスクは、サービステンプレートの［デバッグ］ビューと［タスク］－［デバッグ］ビューに表示されます。Admin

または Develop ロールが割り当てられたユーザーだけがデバッグタスクを参照および使用できます。デバッグタスクはタスクのサマリーに表示されないことに注意してください。

メモ　デバッガービューからサービステンプレートまたは部品の定義を編集することはできません。その代わりに、デバッグ中に欠陥が見つかった場合は、デバッグを中止して［Service Builder Edit］画面に戻り、必要な修正を行う必要があります。

同時のデバッグ操作と関連付けられたタスクの制限

同じサービステンプレートに対して生成されるデバッグサービスおよびデバッグタスクは最大で 1 つです。また、同時に複数のユーザーが同じサービステンプレートをデバッグすることはできません。同じサービステンプレートを複数のユーザーが同時に編集することはできません（最後のテンプレート保存操作が優先されるため、複数の問題を並行して修正することができません）。

デバッグ中のサービステンプレートにデバッグサービスとデバッグタスクがすでに存在している場合、ビルド操作またはリリース操作を実行すると、デバッグサービスとデバッグタスクが自動的に削除され、デバッグを再開したときに新しいデバッグサービスとデバッグタスクが作成されます。

デバッグタスクの終了後、［デバッグ］ビューを閉じずにデバッグを再実行すると、作成されたデバッグサービスと完了したデバッグタスクが自動的に削除され、新しいデバッグサービスとデバッグタスクが生成されます。

前提条件

デバッグするサービステンプレートの開発バージョンは、ビルドプロセスを正常に完了している必要があります。

デバッグセッションを開始するには以下の手順に従います。

操作手順

1. ［デバッグ実行］ダイアログで、適切な詳細情報を入力します。
2. ［サービス名］フィールドで、元の名前に「DEBUG」が付加された名前をそのまま使用するか、他の名前を指定します。
3. 他のサービスおよびタスク関連のフィールドをよく見て、必要であれば修正します。
4. ［ログ出力レベル］で、タスクのログファイルに保存される詳細のレベルを指定します。
5. 必要なら、通常はサービスの構成とサブミットを行うユーザーが指定する値を、ここで指定できます。［編集］プルダウンメニューをクリックし、［［サービス作成］画面］または［［サービス実行］画面］画面を選択します。
6. ［OK］をクリックして、デバッガーのユーザーインタフェースへアクセスします。

操作結果

デバッグインタフェースが開き、ここでデバッグ処理を開始できます。

関連参照

- 6.5 ［デバッグ実行］ダイアログ

6.5 ［デバッグ実行］ダイアログ

サービステンプレートのビルドを正しく行った後で、［デバッグ実行］ダイアログでデバッグセッションに関連する詳細を指定できます。

次の表で、［デバッグ実行］ダイアログのフィールド、サブフィールド、およびフィールドグループについて説明します。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 33 ［デバッグ実行］ダイアログ

フィールド	サブフィールド	説明
［サービス名：］*	-	サービステンプレートに割り当てられた名前を指定します。デバッグバージョンの名前には「DEBUG」が付加されます。
［タグ：］	-	サービスが関連付けられたタググループを指定します。
［タスク名：］	-	割り当てられたタスク名を指定します。デバッグバージョンの名前には「DEBUG」が付加されます。
［タスクの説明：］	-	タスクに任意の短い説明を提供します。
［サービスグループ］：*	-	タスクが関連付けられたサービスグループを指定します。デバッグタスクが操作する対象機器は、エージェントレス接続先定義に影響するため、サービスグループを指定する必要があります。
［ログ出力レベル：］	-	デバッグ処理でタスクのログに出力される情報のレベルを指定します。 • ［0］ • ［10］ • ［20］ • ［30］ • ［40］（デフォルト値）
［プロパティ一覧］ プロパティのリストに以下の詳細が提供されます。 • プロパティ名 • プロパティキー • プロパティ値 • 説明 • スコープ	［編集］	Modify ユーザーおよび Submit ユーザーの視点でサービスを編集することができます。 • ［［サービス作成］画面］ • ［［サービス実行］画面］
	［デフォルトに戻す］	サービステンプレートのビルド時にプロパティ値を復元します。
	［インポート］	プロパティファイルをインポートします。これは別のサービステンプレートから既知の設定を持つプロパティファイル、またはデバッグ中の現在のサービステンプレートのある時点で保存されたプロパティファイルをインポートするのに便利です。
	［エクスポート］	プロパティファイルをエクスポートします。これは特定のサービステンプレートのプロパティを変更を加える前にファイルに保存するのに便利です。これにより、デバッグプロセス中のある時点で必要になった場合に、元のプロパティ設定をインポートすることができます。

関連タスク

- 6.4 デバッガーを実行する
- 6.7 デバッガーで作業する
- 6.6 デバッグ中にサービスエントリと要求エントリを編集する

6.6 デバッグ中にサービスエントリと要求エントリを編集する

［デバッグ実行］ダイアログでデバッグ用のサービスとタスクの詳細を指定できるほか、［サービス作成］または［サービス実行］画面でサービスのプロパティを確認および編集して、構成の指定や、値のサブミットを実行できます。

プロパティセクションには、サービステンプレートに対して実装されるサービスプロパティが表示されます。ここには、［プロパティ名］、［プロパティキー］、［デフォルト値］、［説明］、［スコープ］が表示されます。

多くの場合、デバッガーインタフェースに進む前にまず値を指定する必要があります。この値は、通常はサービスの実行時に Modify および Submit ユーザーにより指定されます。以下のメッセージは、デバッグセッションを続行する前に、指定されたプロパティの値を提供する必要があることを示します。

プロパティ値が不正です。［サービス作成］画面で、プロパティ値を確認してください。

プロパティ値が不正です。［サービス実行］画面で、プロパティ値を確認してください。

［サービス作成］または［サービス実行］画面からプロパティを編集するには、以下の手順に従います。

1. ［デバッグ実行］ダイアログの［プロパティ一覧］セクションで、［編集］をクリックします。
2. 編集するインタフェースに応じて、［［サービス作成］画面］または［［サービス実行］画面］を選択します。
3. 設定のカテゴリ（赤でマークされ、警告（!）アイコンが付いているもの）をクリックし、赤色のアスタリスク（*）でマークされているすべてのフィールドに、欠けている値を入力します。すべての必須フィールドに入力が行われると、セクションが青色に変化します。次に、［OK］をクリックします。
4. デバッグセッション用のサービスとタスクの詳細を指定し、プロパティの必須の値をすべて入力してから、［OK］をクリックして、デバッガーインタフェースにアクセスします。ここでは、サービステンプレートのデバッグを開始できます。

サービスプロパティの編集時に、［インポート］ボタンをクリックして以前のデバッグセッションからプロパティを復元したり、［エクスポート］ボタンで現在のセッションからプロパティを保存したりすることもできます。場合によっては、このインポート機能を使用し、すべてのプロパティ値を一度にロードすることで時間を節約することをお勧めします。また、以前にエクスポートしたプロパティ値を復元することもできます。また、［デフォルト値に戻す］をクリックすると、すべてのデフォルト値を復元できます。

6.7 デバッガーで作業する

サービステンプレートをリリース準備のため正しくビルドし、［デバッグ実行］ダイアログで必要なサービスとタスクの詳細を入力してから、［デバッグ］インタフェースを使用してサービステンプレートをデバッグできます。

［Service Builder Debug］画面には、次の操作パネルが含まれています。

- ［デバッグ］：このエリアでは、現在選択されているステップに関連付けられているデバッグ操作をコントロールできます。
- ［フロー］：サービステンプレートに関連付けられているステップの配置とフローが表示されます。

- ［フローツリー］：ステップの階層フローが表示されます。
- ［タスクログ］：このタブを選択すると、タスクエントリのリストが表示されます。リストを更新して、実行中タスクの現在の状態を確認でき、後で参照するためにリストをファイルへダウンロードすることもできます。
- ［サービスプロパティ］：このタブを選択すると、サービステンプレートに関連付けられているサービスプロパティの［プロパティ名］と、割り当てられている［プロパティキー］および［プロパティ値］が表示されます。
- ［ブレークポイント］：このタブを選択すると、現在設定されているブレークポイントと、その［フロー階層］および［ステップ名］が表示されます。［すべての設定を解除］をクリックすると、現在設定されているすべてのブレークポイントを削除できます。

デバッガー

デバッガーは、デバッグ操作のためタスクの実行をコントロールする便利な方法です。次のオプションを使用できます。

アイコン	オプション
	［応答入力］：待機中状態のタスクについて、入力応答を要求します。
	［(再実行)］：次のようにデバッグを再試行します。 • ［最初からリトライ］：サービステンプレートの先頭からデバッグプロセスを再試行します。 • ［失敗したステップからリトライ］：最後に失敗したステップからタスクを再試行します。 • ［失敗した次のステップからリトライ］：最後に失敗したステップの直後からタスクを再試行します。
	［(再開)］：最後に失敗したステップからデバッグ操作を再開します。
	［(一時停止)］：現在のタスクで、デバッグ操作を中断します。
	［(強制停止)］：現在のタスクで、デバッグ操作を強制的に停止します。
	［(ステップイン)］：中断が可能な次のステップまでタスクを実行します。
	［(ステップオーバー)］：次のステップに含まれる最初の中断可能なポイントまでタスクを実行します。
	［(ステップリターン)］：上位階層に含まれる最初の中断可能なポイントまでタスクを実行します。
	［(ブレークポイント切り替え)］：ブレークポイントを設定または削除します。このブレークポイントにより指定されたステップの後で、タスクの実行が一時中断します。

メモ　システムにより生成されるブレークポイントは、デバッグセッション中にタスクの実行をコントロールするためステップイン、ステップオーバー、またはステップリターンオプションを選択したときに、システムにより自動的に設定されます。

［実行設定］

- ［部品の処理を実行する］：実行モードで部品を実行します。
- ［部品の処理を実行しない］：ドライランモードで部品を実行します。

［タスクの状態］

デバッグタスクの状態は次のいずれかです。

アイコン	状態
	[待機中]：タスクが実行待ちであることを示します。
	[中断中]：ステップ実行機能によりタスクの実行が中断されていることを示します。
	[実行中]：デバッグタスクが進行中であることを示します。
	[応答待ち中]：タスクがユーザーの入力待ちであることを示します。
	[異常検出]：タスクで処理エラーが検出されたことを示します。
	[停止中]：タスクが動作停止または強制動作停止の命令を受信し、処理を中止していることを示します。
	[正常終了]：タスクの実行が正常に完了したことを示します。
	[失敗]：デバッグタスクの実行が失敗したことを示します。

[次のステップフローでの条件式]

次のステップフローで条件式を使用する場合、実行フローを示す行に、現在の状態を示す条件式

 アイコンが表示されます。

- 緑色のアイコン：TRUE
- 濃い灰色のアイコン：FALSE
- 濃い灰色のアイコン：NOT YET（まだ実行されていない）

条件式のアイコンが付いた矢印にマウスを合わせると、条件名、状態、条件式、および説明がツールチップとして表示されます。

ステップ情報

ステップ情報には、与えられたステップについて次のような詳細が表示されます。

- [ステップ ID]：現在選択されているステップの名前が表示されます。
- [ステップ名]：ステップの表示名が表示されます。
- [状態]：現在選択されているステップのデバッグ動作の状態が表示されます。

ステッププロパティ

[ステッププロパティ] ビューには、与えられたステップに関連付けられている入力および出力プロパティと、プロパティの現在の値が表示されます。

次のアイコンは、プロパティの種別を示します。

アイコン	種別
	出力プロパティを示します。

アイコン	種別
	入力プロパティを示します。

プロパティ値を直接編集するには、鉛筆のアイコンをクリックします。[ステッププロパティ編集]ダイアログが表示され、必要な変更を加えることができます。

ステップのプロパティ値を調べるとき、[編集] ボタンをクリックし、[[サービス作成]画面] または [[サービス実行]画面] を選択すると、通常は Modify および Submit ユーザーにより指定されるプロパティの値を指定できます。

また、[インポート] または [エクスポート] ボタンをクリックして、プロパティの値を指定のファイルに保存する、またはファイルから取得することもできます。

[タスクログ]、[サービスプロパティ]、および [ブレークポイント] タブには、現在のデバッグセッションについて有用な追加情報が表示されます。

関連参照

- 6.8 デバッグの詳細情報を調べる

6.8 デバッグの詳細情報を調べる

サービステンプレートのデバッグ中に、実行したタスクに関する詳細、サービスの現在のプロパティ、および設定したブレークポイントを確認できます。

デバッグセッション中に、次のタブから、関連する詳細にアクセスできます。

タスクログ

[タスクログ] タブをクリックすると、現在のデバッグセッション中に実行されたタスクの一覧を確認できます。最新のタスク動作でタスクの一覧を更新するには、[更新] をクリックします。または、[自動更新する] チェックボックスをオンにすると、タスクの一覧が自動的に更新されるようになります。

デバッグセッションで実行されたタスクの記録を保持しておくには、[ダウンロード] をクリックしてから、ログファイルの場所を指定します。デバッガーを開始するときにログファイルの詳細レベルを指定します。

サービスプロパティ

[サービスプロパティ] タブをクリックすると、サービステンプレートに関連付けられているすべてのサービスプロパティを参照できます。これは、特定のステップに関連付けられた、入力と出力のプロパティおよびサービステンプレートのサービスプロパティを確認するのに役立ちます。次のフィールドが提供されます。

- [プロパティ名]：サービスプロパティに割り当てられた表示名。
- [プロパティキー]：サービスプロパティに割り当てられたキー名。
- [プロパティ値]：選択したサービスプロパティに現在割り当てられている値。

ブレークポイント

現在のデバッグセッションで設定されているすべてのブレークポイントを、[ブレークポイント] タブで参照できます。次の詳細が表示されます。

- ［フロー階層］：フローの階層で、通常は円記号（¥）で示されます。
- ［ステップ名］：ステップに割り当てられた表示名。

ステップを選択し、ブレークポイントのアイコンコントロールをクリックすることで、特定のステップで実行フローを一時停止するためのブレークポイントを設定できます。

ブレークポイントが不要になった場合は、［すべての設定を解除］をクリックして、現在設定されているすべてのブレークポイントを削除できます。

6.9 デバッグ中にタスクを管理する

デバッグプロセス中のタスクの実行とフローを制御できます。

［デバッグ］ビューで利用可能なオプションの詳細については、以下のトピックを参照してください。

6.9.1 デバッグタスクの処理フローを制御する

サービステンプレートをデバッグする一般的な手順を以下で説明します。

前提条件
開発状態のサービステンプレートが存在し、［フロー］ビューにステップが追加されている必要があります。

操作手順

1. サービステンプレート内の部品を実行するときに問題の発生を想定していない場合は、デバッグプロセスの第 1 ステップはステップ間で停止させずにデバッグタスクを実行します。［デバッグ］ビュー、またはサービステンプレートのデバッグビューで、部品のフロー遷移や処理に問題がないことを確認します。サービステンプレートにこの時点では実行が好ましくない部品が含まれている場合は、このステップをスキップしてステップ 2 へ進んでください。
2. 部品のフロー遷移または処理で問題を見つけた場合は、デバッグタスクのステップを 1 つずつ実行して、問題が発生している正確な場所と問題の種類を識別してください。入力プロパティと出力プロパティに想定外の値を割り当てることで、部品の動作もテストできます。
   - ［ステップイン］アイコンをクリックして、現在選択されているステップを実行します。
   - ［ステップオーバー］アイコンをクリックして、次のステップを実行します。
   - ［ステップリターン］アイコンをクリックして、現在のフローのステップ実行が完了した後で、上位フローのステップを実行します。
   - デバッグ矢印をクリックし、［失敗したステップからリトライ］を選択して、失敗したステップからデバッグ処理を再実行します。失敗ステップからリトライすることで、同じタスク識別子と元のプロパティ値でデバッグタスクを再開できます。失敗の原因が判明した場合に、このアプローチを利用できます。例えば、ネットワークの一時的な障害で失敗するステップは、ネットワーク接続が再度利用できるようになった時点でリトライできます。
   - デバッグ矢印をクリックし、［失敗した次のステップからリトライ］を選択して、失敗したステップの次のステップからデバッグ処理を再実行します。失敗ステップの後のステップからリトライすることで、同じタスク識別子と元のプロパティ値でデバッグタスクを再開できます。このアプローチが適切となるのは、失敗ステップを実行する必要がない状況です。失敗ステップの後のステップからタスクをリトライすると、タスクの処理は失敗ステップが正常終了していたかのように続いていきます。このアプローチは、あるステップで問題に遭遇しても、いったんデバッグタスクの実行を継続して、問題については後から対処する場合に利用します。

6.9.2 デバッグタスクの割り込みを処理する

状況によっては、デバッグセッション中にタスクの実行が予期せず中断または終了する場合があります。

デバッグタスクのステップは、サービステンプレートのデバッグビューの［フロー］ビューに、実行順で表示されます。各ステップのアイコンは各ステップの状態を示します。デバッグプロセスの中断時にタスクにどのような影響があったかを認識する必要があります。

Ops Center Automator サーバが停止した場合にデバッグタスクを処理する

デバッグ中に Ops Center Automator サーバが停止した場合、実行中のデバッグタスクは強制的に停止されます。したがって、Ops Center Automator サーバを停止する前に、依然として実行されている（まだ完了していない）すべてのデバッグタスクが終了するまで待つか、デバッグタスクを停止してください。これは、通常のサービスの実行時に生成されるタスクの処理と同じです。

クラスタのフェイルオーバーが発生した場合にデバッグタスクを処理する

デバッグ中にクラスタのフェイルオーバー（システムの切り替え）が発生した場合、デバッグタスクは強制的に停止されます。これは、通常のサービスの実行時に生成されるタスクの処理と同じです。

デバッグ中にユーザーがログアウトした場合にデバッグタスクを処理する

デバッグ中に明示的なログアウト操作を行った場合、確認ダイアログが表示され、ログアウトの前に確認するよう求められます。ログアウトを選択すると､デバッグタスクは強制的に停止されます。

ブラウザを閉じた場合にデバッグタスクを処理する

デバッグ中にブラウザを閉じても、デバッグタスクはそのまま実行を続けます。タスクを 1 ステップずつ実行している場合、タスクは停止したステップでとどまります。デバッグタスクを停止するには、再度ログインし、［タスク一覧］ビューからデバッグタスクを停止する必要があります。

6.9.3 ［タスク一覧］へのタスク表示を制御する

タスクをデバッグ中は、［タスク一覧］を使ってタスクの一覧を管理する方法を指定できます。

タスクを一覧に表示しておく期間を指定でき、これを過ぎるとタスクはアーカイブされて［タスク一覧］に表示されなくなります。タスクがアーカイブされる（履歴エントリになる）と、そのタスクについての詳細情報は削除され、［タスク一覧］ビューに戻すことはできません。

次のプロパティの 1 つを使って適切な値を指定することで、タスクのアーカイブ処理を自動的に行わせることができます。

プロパティキー	説明	値の範囲
tasklist.autoarchive.taskRemainingPeriod	日数で表わし、完了したタスクを［タスク一覧］内に保持する期間を指定します。指定した期間が過ぎると、タスクは自動的にアーカイブされます。タスクの自動アーカイブ処理は **tasklist.autoarchive.executeTime** プロパティを通じて指定した時間に従って 1 日に 1 回実施されます。	1～90 デフォルト値：7
tasklist.debugger.autodelete.taskRemainingPeriod	日数で表わし、デバッグタスクが自動的に削除されるまでの期間を指定します。	1～90 デフォルト値：7
tasklist.autoarchive.maxTasks	［タスク一覧］に保持されるタスクとデバッグタスクの最大合計数を指定します。［タスク一覧］	100～5000 デフォルト値：5000

プロパティキー	説明	値の範囲
	内のタスクが最大数を超えると、余分なタスクは終了日付と時間が最も古いものから自動的にアーカイブされます。アーカイブされたタスクは履歴エントリとして管理されます。デバッグタスクは自動的に削除され、履歴には保持されません。自動アーカイブ処理と自動削除は **tasklist.autoarchive.executeTime** プロパティで指定した時間に 1 日に 1 回実施されます。 指定した値よりもタスク数が多い場合は、新しいサービスを実行する試みは「最大値超過」エラーとなり、タスクは生成されません。1 回実行された定期的なタスクはこの制限の対象とならないため新しいタスクを生成できます。したがって、新しいサービスを実行させるには、**tasklist.autoarchive.taskRemainingPeriod** プロパティを指定するために、1 日あたりの実行回数を見積もる必要があります。	

6.10 部品のプロパティマッピングをチェックする

サービステンプレートをデバッグする際に、タスクの流れをどのように実行するかを制御できます。

次に示すのは部品のプロパティマッピングをチェックするための一般的な手順です。

前提条件

開発状態のサービステンプレートが存在し、[フロー] ビューにステップが追加されている必要があります。

操作手順

1. [フロー] ビューで、部品プロパティの値をチェックするステップを選択します。[デバッグ] ビューに、選択したステップの入力プロパティと出力プロパティが表示されます。
2. サービステンプレートの [デバッグ] ビューの下部にある、[サービスプロパティ] タブをクリックします。サービスプロパティの値が表示されます。
3. [デバッグ] ビューで、[ステッププロパティ] の内容を調べて、チェックする部品プロパティを確認し、マッピング先のサービスプロパティを特定します。
4. [サービスプロパティ] タブの [コンポーネント ID] 列から、ステップ 3 で特定したサービスプロパティを見つけます。
5. [デバッグ] ビューと [サービスプロパティ] ビューで、部品のプロパティとマッピングされたサービスプロパティの [値] 列に表示される値が同じであることを確認します。サービスプロパティが意図していた部品のプロパティにマッピングされていない、または部品プロパティとサービスプロパティの値が異なっている場合は、サービステンプレートの [編集] ビューで鉛筆アイコンをクリックし、[ステッププロパティ編集] ダイアログで正しい値を入力して、問題を修正します。部品プロパティの値を変更することもできます。こうすることで、プロパティマッピングが正しく構成されているときに部品 処理をテストして、値をいろいろと変えると引き続くステップの処理およびフローの遷移がどのように変化するかを知ることができます。

関連参照

- 6.11 [ステッププロパティ編集] ダイアログ

6.11 ［ステッププロパティ編集］ダイアログ

デバッグ処理中に、必要に応じてステッププロパティを編集できます。

次の表は、［ステッププロパティ編集］ダイアログのフィールド、サブフィールド、およびフィールドグループについての説明です。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 34 ［ステッププロパティ編集］ダイアログ

フィールド	サブフィールド	説明
［プロパティキー］	-	プロパティ名を表示します。
［プロパティ名］	-	プロパティの表示名を表示します。
［説明］	-	プロパティの説明を表示します。
［複数行］	はい／いいえ	プロパティに対して複数行の値が必要かどうかを示します。
［プロパティ値］*	-	プロパティの現在の値を表示します。

関連タスク

- 6.10 部品のプロパティマッピングをチェックする

6.12 プロパティの値をインポートする

サービステンプレートのデバッグ中に、指定したプロパティファイルにプロパティ値をインポートできます。

以下の手順に従って、指定したファイルにプロパティ値をインポートします。

操作手順

1. デバッガーインタフェースで、［インポート］ボタンをクリックします。
2. ［インポート］ダイアログで、プロパティ値を格納するプロパティファイルの名前を入力するか、ブラウザを使用して目的のファイルを検索し、［OK］をクリックします。
   インポートが完了すると、インポートされたプロパティ値とインポートされなかったプロパティ値に関する通知が次のように一時的に表示されます。
   - 値が適用されたプロパティ。
   - 値が適用されなかったプロパティ。このプロパティは、その属性値を変更できなかったものと、プロパティ値の定義のために値が適用されなかったものです。
   - 存在しないプロパティ。これは、ファイルに定義されており、対象のサービスに存在しないプロパティです。

   デバッガー（または［サービス作成］、［サービス編集］、および［サービス実行］画面）からプロパティファイルをインポートする場合、JSON 形式または key=value 形式がサポートされます。
   CLI からインポート機能にアクセスする場合、key@FILE=ファイルパス形式もサポートされます。

インポート中にプロパティ値を適用するには、以下の表に示した条件を満たす必要があります。

プロパティグループ属性	プロパティ属性				
hidden	paramMode	visibility	reference	hidden	readOnly
false	in	config	false	false	false
false	in	exec	false	--	--

プロパティがこれらの条件を満たさない場合、または対応するプロパティがサービスに定義されていない場合、定義ファイルに含まれる値は適用されません。また、「value」フィールドが定義されていない場合や、「value」フィールドが null に設定されている場合も、値は適用されません。

メモ keyName の長さが制限を超えている場合、そのプロパティは、サービスに存在しないプロパティとして分類されます。

インポート中にエラーが発生した場合、エラーダイアログが表示され、インポートが取り消されるため、すべてのプロパティ値が変更されないまま残ります。指定したファイルが存在しない場合、またはプロパティファイルの定義が無効な場合、エラーが発生します。

関連参照

- 6.13 プロパティの値をエクスポートする

6.13 プロパティの値をエクスポートする

サービステンプレートのデバッグ中に、プロパティファイルをエクスポートできます。

デバッグ中に、プロパティ値をプロパティファイルにエクスポートできます。これにより、複数のプロパティ値をファイルに保存し、後で参照することができます。

プロパティの値をエクスポートする

以下の手順に従って、指定したファイルにプロパティ値をエクスポートします。

1. デバッガーインタフェースで、[エクスポート] ボタンをクリックします。
2. ブラウザーを使用してプロパティファイルを見つけるか、名前を直接入力して、[OK] をクリックします。

指定したファイルにプロパティ値がエクスポートされます。プロパティ値は JSON 形式でエクスポートされ、デフォルトで service_properties.json ファイルに保存されます。

形式

コマンド引数として指定したプロパティファイル内で、実行するサービスで使用するプロパティキーと値を JSON 形式、key=value 形式、および key@FILE=ファイルパス形式で定義できます。

JSON 形式 :

```
{
"properties": [
      {
        "keyName": "プロパティキー",
     "displayName" : "プロパティ表示名"
     "description" : "プロパティの説明"
     "type" : "プロパティのデータ型"
```

```
        "value": "プロパティ値"
      },
      {
        "keyName": "プロパティキー",
     "displayName" : "プロパティ表示名"
     "description" : "プロパティの説明"
     "type" : "プロパティのデータ型"
        "value": "プロパティ値"
      },
...
]
}
```

JSON 形式の定義の詳細を以下に示します。

- 「displayName」、「description」、および「type」フィールドの指定は任意です。
- 「value」フィールドを指定した場合、プロパティ値には空の値を設定します。
- パスワードタイププロパティの値は、プレーンテキストまたは暗号化形式で指定できます。セキュリティ上の理由で、パスワードタイププロパティの「value」フィールドはエクスポートされません。定義した値はそのままインポートされます。プレーンテキストまたは暗号化形式のどちらであるかは REST API が判断します。
- 定義ファイル内では、値を設定するプロパティのみを定義します。インポートしたファイル内で定義されていないプロパティの値はそのまま残ります。ステッププロパティをエクスポートする場合、「type」フィールドはサービスコンポーネントに対してのみ出力されます。

key=value 形式 :

key=value プロパティファイルのプロパティ値を指定するには、次の形式を使用します。

```
property-key=プロパティ値 [改行]
```

key=value 形式の定義の詳細を以下に示します。

- 各行でプロパティ名とプロパティ値を指定します。
- 番号記号（#）で始まる行はコメント行として処理されます。
- 等号（=）が含まれない行はコメント行として処理されます。
- 各プロパティ設定行の末尾に改行を追加する必要があります。
- プロパティ名とプロパティ値の行の途中に改行を追加しないでください。
- 大文字と小文字は区別されます。
- サービスや部品のリソースファイルのように「¥」が文字列に含まれる場合でも、「¥¥」と入力する必要はありません。
- 「¥」は「¥」として処理されます。
- 行の先頭から最初の等号（=）までの文字がプロパティ名として扱われます。プロパティ設定行の前後で行を切り取らないでください。
- プロパティ名の後の等号以降、行の末尾までの文字はプロパティ値として扱われます。プロパティ設定行の前後で行を切り取らないでください。
- プロパティファイルの最後（EOF）の行末文字の指定は任意です。
- 空の行（改行のみを含む行）は無視されます。
- CR+LF と LF を改行として使用できます。
- 「property-key = [改行]」形式を使用した場合、プロパティ値に空の値を設定します。

key@FILE=ファイルパス形式：

この形式では、プロパティキーがプロパティファイルに格納され、プロパティ値がプロパティ値ファイルに格納されるため、プロパティ値は単独で参照されます。プロパティファイルでは、この形式と key=value 形式を併用できます。

```
property-key@FILE=プロパティ値ファイルの絶対パスまたはプロパティファイルを基点とした相対パス ［改行］
```

key@FILE=ファイルパス形式の定義は、key=value 形式と同じです。key=value 形式との違いを以下に示します。

- プロパティ値ファイルの絶対パスまたはプロパティファイルからの相対パスを常に指定する必要があります。指定しない場合、エラーが発生します。
- 指定したファイルにプロパティ値が含まれない場合、エラーが発生します。
- プロパティ値ファイルには、行末文字を含めることができます。ただし、行末文字を設定してはならないプロパティに対して、行末文字を含むプロパティ値ファイルを指定した場合、エラーが発生します。

関連参照

- 6.12 プロパティの値をインポートする

6.14 サービステンプレートをリリースする

サービステンプレートを完成させるための最後の手順はリリースプロセスです。リリース操作を行うと、サービステンプレート、および関連する部品の構成タイプが［リリース］に変更されます。新規のサービスおよびタスクは、リリース状態のサービステンプレートから作成できます。

リリース状態のサービステンプレートは、サービスの作成元として使用できるサービステンプレートとして Ops Center Automator に表示されます。

メモ
リリース操作は 1 回だけ実行できます。

メモ
リリースされたサービステンプレートは、サービステンプレートの開発中にサービス部品としてフローに配置することもできます。
サービス部品を使用したステップの戻り値は、以下のとおりです。
0：サービス部品内のステップが正常に終了した。
1：サービス部品内のステップが警告終了した。異常終了したステップはなし。
2：サービス部品内のステップが異常終了した。

前提条件
エラーなしでビルドプロセスが完了し、設計通りに実行される開発状態のサービステンプレート。

操作手順

1. ［Service Builder Edit］画面で、［リリース］をクリックします。
   サービステンプレートに対してリリースプロセスが実行され、構成タイプがリリース状態に変更されます。

操作結果

リリースプロセスが正常に完了すると、サービステンプレートの構成タイプがリリースに変更され、［サービス］タブで利用可能になります。

- そのサービステンプレートは開発状態から削除され、利用可能なサービステンプレートを参照するとき［リリース］タブに表示されるようになります。
- テンプレートが開発状態であるときに作成されたサービスは削除されます。
- 開発状態のときにテンプレートから実行されたタスクはすべてアーカイブされます。
- サービステンプレートは、新しいサービスを作成すると［サービステンプレート］リストに表示されます。
- 関連する部品は、［コンポーネント］ビューの［リリース］タブに表示されます。
- サービスの定義で指定されている場合、サービスコンポーネントが作成され、［コンポーネント］ビューの［サービス］タブに配置されます。

次の作業

リリース済みのサービステンプレートを使用して新しいサービスを作成します。サービスの作成に関する詳細情報については、『Hitachi Ops Center Automator ユーザーズガイド』を参照してください。

関連参照

- 6.3 ［ビルド/リリース結果］ダイアログ

7

高度なオプション

ここでは、サービステンプレートと部品の管理に利用可能な他の機能を説明します。

7.1 サービステンプレートの属性を編集する

［サービス定義編集］ダイアログでは、サービステンプレートに関連付けられている詳細を参照およびカスタマイズでき、サービステンプレートの表示方法と実行のスケジューリングに影響を及ぼすカスタムファイルを指定できます。

前提条件

サービステンプレートの状態はリリースまたは開発である必要があります。

操作手順

1. ［Service Builder Edit］画面の［定義情報］タブで、［編集］ボタンをクリックします。
   ［サービス定義編集］ダイアログが表示されます。
2. サービステンプレートの情報を入力し、［OK］をクリックします。
   ［サービス定義編集］ダイアログが閉じます。

操作結果

指定したすべての変更が、更新したサービステンプレートに反映されます。

次の作業

リストから各部品を選択し、サービステンプレートの編集を続け、入力と出力のステッププロパティを指定してフローを作成します。

関連参照

- 7.2 ［サービス定義編集］ダイアログ

7.2 ［サービス定義編集］ダイアログ

サービステンプレートを編集または作成するときに、サービステンプレートの追加の詳細を表示および指定できます。

次の表で、［サービス定義編集］ダイアログのフィールド、サブフィールド、およびフィールドグループについて説明します。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 35 ［サービス定義編集］ダイアログ

フィールド	サブフィールド	説明
［サービステンプレート ID］	［-］	サービステンプレートのキー名。
［サービステンプレートバージョン］	［-］	サービステンプレートのバージョン。
［ベンダー ID］	［-］	サービステンプレートのベンダー ID。
［サービステンプレート名］*	［-］	ユーザーに表示するサービステンプレート名。

フィールド	サブフィールド	説明
［ベンダー名］	［-］	サービステンプレートのベンダー名。
［説明］	［-］	サービステンプレートの説明。
［タグ］	［-］	サービステンプレートに関連付けられるタグ。
［詳細設定］	［アイコン］	PNG 形式の画像ファイル（48 x 48 ピクセル）。デフォルトの画像が用意されていますが、［変更］ボタンをクリックするとアイコンを変更できます。［デフォルトアイコンを指定］ボタンをクリックすると、デフォルトのアイコンに戻すことができます。
	［画面カスタムファイル］	カスタムファイルでは、［詳細設定］ダイアログに表示する情報と、サービスに関連付けられた概要が提供されます。.html、.js、.css、.jpeg 形式のファイルをアップロードするには、［選択］をクリックします。何かの理由で、テンプレートに関連付けられているカスタムファイルを削除する場合は、［削除］をクリックします。
	［サービス詳細説明画面のファイル名(相対パス)］	［詳細設定］ダイアログに関する情報を提供するファイルの名前を指定します。
	［サービス概要のファイル名(相対パス)］	ユーザーに表示するサービステンプレートに関連付けられた概要の情報が含まれるファイルの名前を指定します。
	［スケジュール種別：］	タスクを実行するスケジュールは、［即時実行］、［指定日時実行］、または［定期実行］です。
	［有効なアクション:］	ステップの処理が何らかの理由でハングアップした場合、サービスの実行の［強制停止］をユーザーに許可するか、失敗したステップまたは最後に失敗したステップの直後の時点からサービステンプレートの処理の［リトライ］をユーザーに許可するかを選択できます。

アスタリスク（*）付きのフィールドは必ず指定します。

関連タスク

- 7.1 サービステンプレートの属性を編集する

7.3 プロパティグループを作成する

プロパティグループを作成して、サービステンプレートに関連付けられたプロパティを分類できます。

サービステンプレートの入力および出力プロパティは、カスタムのプロパティグループに割り当てることができます。プロパティグループが必要ない場合は、デフォルト値の［reserved.defaultGroup］が使用されます。プロパティグループを作成すると、入力および出力プロパティが、それぞれが割り当てられているカスタムプロパティグループの名前に従って表示されます。

前提条件
開発状態のサービステンプレートが存在している必要があります。

操作手順

1. ［Service Builder Edit］画面の［プロパティ］タブで、［追加］プルダウンメニューから［プロパティグループ］オプションを選択します。

［プロパティグループ作成］ダイアログが表示され、新しいプロパティグループの詳細を入力できます。

2. プロパティグループに関連する詳細を入力して、［OK］をクリックします。
   新しく追加したグループがプロパティ一覧に追加されます。
3. プロパティまたは変数を作成したあと、新しく作成したグループに追加することができます。
   既存のプロパティをプロパティ一覧から選択して、適切なグループへ移動することもできます。

関連参照

- 7.3.1 ［プロパティグループ作成］ダイアログ
- 7.3.2 ［プロパティグループ編集］ダイアログ

7.3.1 ［プロパティグループ作成］ダイアログ

サービステンプレート用の新しいプロパティグループは、［プロパティグループ作成］ダイアログから作成できます。

次の表は、［プロパティグループ作成］ダイアログのフィールド、サブフィールド、フィールドグループについての説明です。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 36 ［プロパティグループ作成］ダイアログ

フィールド	サブフィールド	説明
［プロパティグループ ID］*	-	新しいプロパティグループの ID を指定できます。
［プロパティグループ名］*	-	ユーザーインタフェースを通じて示される新しいプロパティグループの名前。
［説明：］	-	新しいプロパティグループ用のオプション説明。
［表示/非表示：］	-	プロパティグループを表示するか非表示にするかを指定します。
［画面カスタムファイル：］	-	プロパティグループ用のカスタムファイルを指定します。

［サマリパネル描画スクリプト：］	-	サマリパネルの内容を表示するための JavaScript 関数を定義します。
		サンプルを次に示します。
		`function summarize(properties, language, displayType) {`
		`  // PropertyInformation objects are stored in the properties.`
		`  var summaryContentsMap = {};`
		`  var restriction;`
		`  for (var i = 0; i < properties.length; i++) {`
		`  restriction = JSON.parse(properties[i].restriction);`
		`  if (displayType == "exec") {`
		`  if (restriction.permission != "hidden") {`
		`  summaryContentsMap[properties[i].displayName] = properties[i].value;`
		`  }`
		`  } else {`
		`  summaryContentsMap[properties[i].displayName] =`

フィールド	サブフィールド	説明
		properties[i].value; } } return summaryContentsMap; }
[バリデーション スクリプト:]	-	プロパティグループの検証ファイルを生成します。

アスタリスク（*）が付いたフィールドは必須です。

プロパティグループへの入力をチェックするための検証スクリプトを作成する

提供されている検証オプションが目的にそぐわない場合には、スクリプトを作成して必要なチェックをすることができます。次に示すのは、ユーザーの入力値が許される最大値の 2048 以下の数字であることをチェックするために JavaScript で書いた検証スクリプトです。

```
function (properties, lang, displayType) {
    var message = [];
       var hasError = false;

    if (displayType == "exec") {
        _.each(properties, function(property) {
            if (isNaN(property.value)) {
                message.push( "value must be a number:"+
property.keyName + "=" + property.value);
                hasError = true;
                }

            if (property.value >= 2048) {
                message.push ("value must be less than 2048:"+
property.keyName + "=" + property.value);
                hasError = true;
            }
        });
    }

    if (hasError) {
        return message;
    } else {
        return
    }

}
```

次の表は入力プロパティ用の検証スクリプトを示します。

#	名前	説明

1	スクリプトフォーマット	function (arg1, arg2, arg3) {
		//コード
		}

2	検証スクリプトの引数	arg1: プロパティグループのプロパティ値のリスト。各要素は次のプロパティを持っているオブジェクトです。 • keyName：プロパティの文字列形式の名前。 • value：プロパティの文字列形式の値。 arg2： ロケール文字列。例：ja、 en arg3:

#	名前	説明
		スクリプト動作中の操作情報（タスク作成操作：exec、プロパティの編集操作：config）
3	検証スクリプトの戻り値	成功： undefined または null 失敗： 配列または文字列形式のエラーメッセージ

値が数値ではないか、または指定最大値よりも大きい場合には、ユーザーインタフェースを通じてメッセージが出力されます。

関連タスク

- 7.3 プロパティグループを作成する

7.3.2 ［プロパティグループ編集］ダイアログ

サービステンプレート用の既存のプロパティグループは、［プロパティグループ編集］ダイアログで編集できます。

次の表は、［プロパティグループ編集］ダイアログのフィールド、サブフィールド、およびフィールドグループについての説明です。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

ダイアログに情報を入力し、その情報が無効である場合、問題の説明を記載したエラーがボックスの横に表示されます。

表 37 ［プロパティグループ編集］ダイアログ

フィールド	サブフィールド	説明
［プロパティグループ ID：］	-	プロパティグループの ID を指定します。
［プロパティグループ名：］	-	ユーザーインタフェースに表示されるプロパティグループの名前。
［説明：］	-	プロパティグループの説明（任意）。
［表示/非表示：］	-	プロパティグループを表示するか非表示にするかを指定します。
［画面カスタムファイル：］	-	プロパティグループ用のカスタムファイルを指定します。
［サマリパネル描画スクリプト：］	-	サマリパネルの内容を表示するための JavaScript 関数を定義します。
［バリデーションスクリプト：］	-	プロパティグループの検証ファイルを指定します。

アスタリスク（*）が付いたフィールドは必須です。

関連タスク

- 7.3 プロパティグループを作成する

7.4 バージョンを管理する

Service Builder では、サービステンプレートと部品の両方に同じバージョン管理方法が適用されます。新しいサービステンプレートまたは部品が作成されると、バージョン番号が割り当てられます。

すべてのサービステンプレートおよび部品には、nn.nn.nn 形式のバージョン番号が必要です（*major-version-number*.*minor-version-number*.*revision-number*）。各数値は 00～99 の範囲で指定します。

同じサービステンプレート ID およびベンダー ID を持つサービステンプレートまたは同じ部品 ID およびベンダー ID を持つ部品を複製する場合、新しい複製には新しいバージョン番号を割り当てる必要があります。同じ部品キー名、ベンダー ID、およびバージョン番号を持つサービステンプレートまたは部品は複製できません。

同じサービステンプレートキー名とベンダー ID を持つサービステンプレートが 2 つ以上存在する場合、サービステンプレートの最新バージョンだけが表示されます。同様に、同じ部品キー名とベンダー ID を持つ部品が 2 つ以上存在する場合、コンポーネントの最新バージョンだけが表示されます。

特定の条件で、部品が更新されると、サービステンプレートが最新バージョンへの更新が必要な古いバージョンになってしまうことがあります。この場合、以下の手順で個々の部品を更新するか、特定のサービステンプレートのすべての部品を更新できます。

1. ［Service Builder Edit］画面で、［操作］プルダウンメニューから［コンポーネントバージョン管理］オプションを選択します。［コンポーネントバージョン管理］ダイアログが表示されます。
2. 現在のサービステンプレートに関連付けられているすべての部品を最新バージョンに更新するには、［一括適用］タブを選択します。［ステップ一覧］をクリックすると、更新対象の部品コンポーネントのステップがすべて表示されます。更新する部品コンポーネントを個別に指定するには、［個別適用］タブを選択し、特定のコンポーネントのステップに適用するバージョンを選択します。
3. 更新するコンポーネントを選択してから［適用］をクリックすると、選択したバージョンが適用されることの確認メッセージが表示されます。
4. バージョンの更新が完了したら、［閉じる］をクリックします。
5. 部品バージョンを更新したら、行った変更を保持するため部品を保存する必要があります。

メモ　サービスを更新する前に、サービステンプレートはリリース状態になっている必要があり、かつサービスに関連付けられたすべての既存のタスクはアーカイブまたは削除されている必要があります。

関連参照

- 7.4.1 ［コンポーネントバージョン管理］ダイアログ

7.4.1 ［コンポーネントバージョン管理］ダイアログ

サービステンプレートと関連したコンポーネントのステップのバージョンを管理して、特定のコンポーネントの最新バージョンを使用していることを保証できます。すべてのコンポーネントを更新することも、特定のコンポーネントを選択することもできます。

次の表で、［コンポーネントバージョン管理］ダイアログのフィールド、サブフィールド、およびフィールドグループについて説明します。*フィールドグループ*は、特定の操作または構成に関連するフィールドの集まりです。

表 38 ［コンポーネントバージョン管理］ダイアログ

フィールド	サブフィールド	説明
［一括適用］	-	すべてのステップコンポーネントをまとめて最新バージョンに更新します。
［個別適用］	-	最新のバージョンに更新する特定のステップコンポーネントを選択的に指定します。このオプションを選ぶと、コンポーネントの一覧が、そのコンポーネントに関連している詳細情報やステップの一覧とともに表示され、コンポーネントを選択して［バージョン］プルダウンメニューから更新を実行できます。コンポーネントのさらに詳しい情報を表示するには［参照］をクリックすると［Service Builder View］画面または［部品参照］ダイアログが表示され、［プロパティ］タブをクリックするとコンポーネントに関連付けられているすべてのプロパティが表示されます。
［ステップ一覧］	-	［ステップ名］、［適用済みコンポーネント名］、［適用対象バージョン］、［適用対象コンポーネント名］、および［状態］を含むステップ一覧を示します。［適用対象コンポーネント名］をクリックすると［コンポーネントプレビュー］ダイアログが表示され、指定したコンポーネントについてさらに詳細情報が表示されます。［コンポーネントプレビュー］ダイアログの［参照］をクリックすると［Service Builder View］画面または［部品参照］ダイアログが表示され、［プロパティ］タブをクリックするとコンポーネントに関連付けられているすべてのプロパティが表示されます。

関連概念

- 7.4 バージョンを管理する

A

参照情報

ここでは、ビルトインサービステンプレート、ビルトイン部品、予約プロパティの一覧や、部品用のロケール設定について説明します。

A.1 ビルトインサービステンプレートの一覧

Service Builder には、設定済みの一連のサービステンプレートが付属します。

次のサービステンプレートがデフォルトで提供されており、追加設定なしでサービスの実行に使用できます。

Add Host to Cluster in vCenter

ESX クラスタホストによってデータストアとして使用される既存ボリュームを新しい ESX ホストに割り当てます。

Allocate Fabric Aware Volumes and Create Datastore for ESX Cluster

VMware ESX クラスタホストにボリュームを割り当て、ゾーニングを構成し、データストアの下に VMware データストアを作成します。

Allocate Fabric Aware Volumes with Configuration Manager

一般的なアプリケーションを実行しているサーバで消費するボリュームを、Configuration Manager を通して、関連するインフラストラクチャーグループから割り当てます。このサービスでは、新しいボリュームをホストに割り当てるときに、FC スイッチ管理製品にアクセスして、既存のファブリック構成とゾーニング情報を取得します。

Allocate Like Volumes with Configuration Manager

指定されたソースボリュームが同じ LUN パスで割り当てられているホストに新しいボリュームを割り当てます。

Allocate Volumes, Fabric, and Datastore for ESXi Host

Configuration Manager を通じて、VMware vSphere サーバ（ESXi ホスト）にボリュームを割り当て、ゾーニングを構成し、VMware データストアを作成します。

Allocate Volumes from Virtual Storage Machine

データ移行用に、仮想ストレージマシンから新しいボリュームを割り当ててほかのストレージシステム上の同じ LDEV ID を予約します。

Allocate Volumes with Clone/Snapshot

in-system replication （Thin Image、 ShadowImage）にて、一般的なアプリケーションを実行している複数のサーバで消費するボリュームを、Configuration Manager を通して、関連するインフラストラクチャーグループから割り当てます。

Allocate Volumes with Configuration Manager

一般的なアプリケーションを実行しているサーバで消費するボリュームを、Configuration Manager を通して、関連するインフラストラクチャーグループから割り当てます。

Allocate Volumes with 2DC Remote Replication

一般的なアプリケーションを実行している複数のサーバで消費するために関連するインフラストラクチャーグループからボリュームを使用してインテリジェントな割り当てを行い、Remote Replication のために新しいコピートポロジーを作成します。

Allocate Volumes with Remote Replication (Global-Active Device)

in-system replication （global-active device）にて、一般的なアプリケーションを実行している複数のサーバで消費するボリュームを、Configuration Manager を通して、関連するインフラストラクチャーグループから割り当てます。

Allocate Volumes with Smart Provisioning

一般的なアプリケーションを実行しているサーバで使用するボリュームを、Configuration Manager を通して、関連するインフラストラクチャーグループからスマートに割り当てます。

Call ServiceNow Table API

ServiceNow の Table API を呼び出します。

Clean up Online Migration Pair

Create Online Migration Pair タスクによって作成されたリソースのクリーンアップができます。

Create High Availability Pair for Migration

データ移行用に、二つのストレージシステム間の仮想ストレージマシンから高可用性ペアを作成します。

Create Online Migration Pair

Configuration Manager を介したオンラインホスト移行のために、ゾーンの作成からコピーペアの作成までを実行します。このサービスが完了したら、Migrate Data for Online Migration Pair サービスを実行して移行を完了する必要があります。

Create ServiceNow Incident Ticket

ServiceNow のインシデントチケットを作成します。

Expand Volume Capacity

既存ボリュームの容量を拡張します。

Export Virtual Storage Machine Configuration Across Sites

Data Mobility サービスの高可用性ペアに関する情報を含む、サイト間の仮想ストレージマシンの構成詳細を示すレポートを提供します。

Global-Active Device Setup

仮想ストレージマシンを作成し、クォーラムディスク ID を割り当て、リモートパスを作成し、コマンドデバイスを割り当てて global-active device を作成します。

Get IO Control

I/O 制御情報を取得します。

Migrate Data for Online Migration Pair

Configuration Manager を介したオンラインホスト移行のために、コピーペアのスワップからソースボリュームの削除までを実行します。このサービスを実行する前に、Create Online Migration Pair サービスが完了している必要があります。

Migrate Data Using High Availability Pair

二つのストレージシステム間の仮想ストレージマシンから高可用性ペアを使用してオンラインでデータ移行できます。

Remove Host from Cluster in vCenter

VMFS データストアをアンマウントし、指定した ESX ホストからボリュームの割り当てを解除し、ゾーニングを削除します。

Set IO Control

WWN または iSCSI 名、および LUN パスを持つボリュームの I/O 制御を設定します。

Remove IO Control

WWN または iSCSI 名、およびボリュームから I/O 制御設定を削除します。

Retrieve ServiceNow Incident Tickets

ServiceNow のインシデントチケットを収集します。

Smart Allocation for Oracle Databases

ストレージシステムにボリュームを割り当て、そして Linux の Oracle ASM のディスクグループにボリュームを追加します。

Update ServiceNow Incident Ticket

ServiceNow のインシデントチケットを更新します。

メモ　これらのサービステンプレートのファイルタイププロパティにはさまざまなコンポーネントの接続の詳細が含まれます。このプロパティを変更しないでください。変更すると、テンプレートが適切に機能しなくなる可能性があります。

警告　ビルトインサービステンプレートに関連付けられる一部のプロパティには、「Do not change」と示されている内部データが含まれます。これらのプロパティを変更しないでください。

A.2 ビルトイン部品の一覧

Ops Center Automator には一連の部品が付属しており、この部品を使用して、カスタマイズしたサービステンプレートを作成できます。デフォルトで含まれる部品を以下の表に示します。

メモ　この表に記載されていない部品は内部使用のみを目的としているため、使用しないでください。

部品名	説明
異常終了部品	実行中のフロー、タスク、階層フロー、および繰り返して実行されるフローの異常終了を管理します。
値判定分岐部品	サービスプロパティの値を比較および判別して処理フローを分岐する条件分岐を作成します。
戻り値判定分岐部品	前のステップの戻り値を比較および判別して処理フローを分岐する条件分岐を作成します。
メール通知部品	SMTP サーバに接続し、指定した宛先にメールを送信します。
ファイル転送部品	相手ホストにファイルまたはフォルダを送信したり、相手ホストからファイルまたはフォルダを受信したりします。転送プロトコルとして SCP または SFTP を使用します。
階層フロー部品	ネストされたいくつかのフローで定義された階層を持つ 1 つのフローを作成します。
汎用コマンド実行部品	相手ホストでコマンドラインを実行します。
実行間隔制御部品	待機時間を指定して、ステップ間の間隔を制御します。
JavaScript 実行部品	JavaScript で記述されたあらゆるスクリプトを実行します。
JavaScript Plug-in for Configuration Manager REST API	JavaScript で書かれたスクリプトを実行し、また、Configuration Manager REST API にアクセスするためのビルトインメソッドを含んでいます。この部品を使用することで、Configuration Manager REST API にアクセスするためのスクリプトを簡単に書くことができます。
Python 実行部品	Ops Center Automator のホスト上で Python スクリプトを実行します。

部品名	説明
繰り返し実行部品	特定のフローを繰り返して実行します。この部品を使用してループ処理を実装できます。
標準出力部品	指定した値を標準出力に出力します。新しいサービステンプレートでこの部品を使用しないでください。この部品は、以前の手順で作成されたサービステンプレートとの互換性を確保するための部品です。
ターミナルコマンド実行部品	ターミナル接続部品を使用して Telnet または SSH で接続されている相手ホストで、コマンドラインを実行します。
ターミナル接続部品	Telnet または SSH 認証を使用した、相手ホストとのターミナル接続を有効にします。
ターミナル切断部品	ターミナル接続部品を使用して Telnet または SSH で接続されているターミナルを相手ホストから切断します。
値判定部品	指定された条件とサービスプロパティの値を比較し、値が条件と一致する場合、0 を返します。
ユーザー応答待ち部品	処理を一時停止して、ユーザーの応答を待機します。この部品を使用すると、オペレータは処理を続行するかどうかを手動で選択します。
Web クライアント部品	HTTP リクエストメッセージ（GET/POST など）を送信し、HTTP レスポンスメッセージを受信します。リクエストされた場合、プロキシ経由でサーバにアクセスし、サーバとプロキシの認証を完了します。例えば、REST API を使用して Web に接続する場合、この部品を使用できます。
ファイルエクスポート部品	入力された内容を指定されたファイルへエクスポートします。

A.3 予約プロパティの一覧

予約プロパティは、Ops Center Automator に特定の定義や目的を持つプロパティキーを持った特殊なサービスプロパティです。

予約プロパティのプロパティキーは「reserved.」で始まります。予約プロパティを使用するには［入力プロパティマッピング］ダイアログまたは［出力プロパティマッピング］ダイアログで、これらを部品のプロパティにマッピングします。ユーザーが予約プロパティに値を定義したり割り当てたりすることはできません。

予約プロパティを入力プロパティにマッピングすると、部品が実行されたときに、予約プロパティの値が部品プロパティに割り当てられます。また、［直接入力］オプションを選択し、［値］フィールドで、「?dna_reserved-property-key?」の形式で予約プロパティを指定する方法もあります。この場合、予約プロパティの値は、部品の実行時に部品プロパティの値の一部を提供します。

予約プロパティを出力プロパティとして使用する場合、予約プロパティには指定した部品プロパティの値が格納されます。［出力プロパティマッピング］ダイアログで［プロパティ参照］オプションを選択すると、出力プロパティを渡す予約プロパティを指定できます。

予約プロパティの一覧を次に示します。

予約プロパティキー	説明
reserved.debugger.exitCode	デバッグモードで実行されたステップの終了コードが格納されます。このプロパティは［デバッグ］画面でのみ使用できます。
reserved.external.path	動的データにアクセスするための REST API のパスを示します。例えば、"http://localhost:22015/Automation/v1/objects/ExternalResources/IPAddresses?host=myHost&type=vm"の場合、値は"/IPAddresses"となります。クエリパラメタは含まれません。
reserved.external.query	REST API に渡されるパスのクエリパラメタを示します。例えば、"http://localhost:22015/Automation/v1/objects/ExternalResources/IPAddresses?host=myHost&type=vm"の場合、値は"host=myHost&type=vm"となります。
reserved.external.resource.dir	動的データの外部リソースのリソースフォルダを示します。
reserved.external.userName	REST API から Ops Center Automator にログインするユーザーの名前です。
reserved.loop.index reserved.loop.indexN	ベースステップより 1 つ（または N）レベル上にある繰り返し実行部品が繰り返された回数を示す 1~99 の数値を参照します。
reserved.loop.input reserved.loop.inputN	ベースステップより 1 つ（または N）レベル上にある繰り返し実行部品の inputProperties 入力プロパティの値を参照します。このプロパティは、繰り返し実行部品の入力プロパティに指定されたコンマ区切りの値のうち、フローの現在の繰り返しに対応する要素の値を格納します。例えば、入力プロパティが「A,B,C」であった場合、値 A、B、C がフローの繰り返しカウントに対応する順序で入力されます。繰り返し実行部品は、最大で 99 回実行されます。
reserved.loop.output	繰り返し実行部品の outputProperties 出力プロパティに値を渡します。このプロパティに出力される値は、コンマ区切りの値として出力プロパティに割り当てられます。例えば、部品の出力プロパティの値が、連続する繰り返しで X、Y、Z であった場合、出力プロパティには「X,Y,Z」が代入されます。
reserved.service.name	タスクの生成元であるサービスの名前を参照します。この予約プロパティを参照するには、「?dna_reserved.service.name?」の形式でプロパティキーを指定します。このプロパティは、サービスプロパティをマップできるすべての部品で使用できます。
reserved.service.serviceGroupName	タスクの生成元であるサービスが登録されているサービスグループを参照します。この予約プロパティを参照するには、「?dna_reserved.service.serviceGroupName?」の形式でプロパティキーを指定します。このプロパティは、サービスプロパティをマップできるすべての部品で使用できます。
reserved.step.path	現在実行中のステップの ID を参照します。この予約プロパティを参照するには、「?dna_reserved.step.path?」の形式でプロパティキーを指定します。このプロパティの値は、部品実行の開始時および終了時にタスクログに出力されるメッセージに表示されるステップ ID と同じです。このプロパティは、サービスプロパティをマップできるすべての部品で使用できます。
reserved.step.prevReturnCode	先行ステップ（部品に接続された関係線の始点であるステップ）の戻り値を提供します。この予約プロパティを参照するには、「?

予約プロパティキー	説明
	dna_reserved.step.prevReturnCode?」の形式でプロパティキーを指定します。先行するステップが複数ある場合、すべての戻り値の論理和がプロパティに割り当てられます。先行するステップがない場合、0 が割り当てられます。このプロパティは、サービスプロパティをマップできるすべての部品で使用できます。先行ステップを実行せずにこの予約プロパティを参照するステップからタスクを再試行した場合、先行ステップが最後に実行されたときの戻り値が、先行ステップの戻り値としてこの予約プロパティに設定されます。
reserved.task.description	タスクの説明を提供します。この予約プロパティを参照するには、「?dna_reserved.task.description?」の形式でプロパティキーを指定します。このプロパティは、サービスプロパティをマップできるすべての部品で使用できます。
reserved.task.dir	タスク実行中に作成された一時データフォルダのパスを提供します。このプロパティは各タスクの実行時に一意のフォルダパスを提供します。このプロパティが参照するフォルダは、タスクが実行されたときに Ops Center Automator サーバ上に作成され、タスクがアーカイブされたときに削除されます。タスクで始まるファイルおよびフォルダは Ops Center Automator で予約されており、ユーザーが作成することはできないことに注意してください。
reserved.task.id	タスク ID を提供します。この予約プロパティを参照するには、「?dna_reserved.task.id?」の形式でプロパティキーを指定します。このプロパティは、サービスプロパティをマップできるすべての部品で使用できます。
reserved.task.name	タスク名を提供します。この予約プロパティを参照するには、「?dna_reserved.task.name?」の形式でプロパティキーを指定します。このプロパティは、サービスプロパティをマップできるすべての部品で使用できます。
reserved.task.submitter	実行するためにタスクをサブミットしたユーザーのユーザー ID を提供します。タスクを再試行した場合、このプロパティはタスクを再試行したユーザーではなく、タスクをサブミットしたユーザーのユーザー ID を参照します。この予約プロパティを参照するには、「?dna_reserved.task.submitter?」の形式でプロパティキーを指定します。このプロパティは、サービスプロパティをマップできるすべての部品で使用できます。
reserved.task.tags	タスクに設定されているタグを参照する予約プロパティです。
reserved.task.url	［タスク詳細］ダイアログにアクセスするための URL を提供します。この予約プロパティを参照するには、「?dna_reserved.task.url?」の形式でプロパティキーを指定します。このプロパティは、サービスプロパティをマップできるすべての部品で使用できます。
reserved.terminal.account	ターミナル接続部品が使用するユーザー ID を参照します。このプロパティは、ターミナルコマンド実行部品の commandLine 入力プロパティによって使用されます。ターミナルに接続するのに使用されたユーザーアカウントのログイン名が格納されます。
reserved.terminal.password	ターミナル接続部品が使用するパスワードを参照します。このプロパティは、ターミナルコマンド実行部品の commandLine 入力プロパ

予約プロパティキー	説明
	ティによって使用されます。ターミナルに接続するのに使用されたユーザーアカウントのパスワードが格納されます。
reserved.terminal.suPassword	ターミナル接続部品が使用する管理者パスワードを参照します。このプロパティは、ターミナルコマンド実行部品の commandLine 入力プロパティによって使用されます。ターミナルに接続するのに使用されたスーパーユーザーのパスワードが格納されます。

A.4 部品用のロケール設定

部品が操作を行う機器に適用されるロケール設定は、オペレーティングシステムによって異なります。以下に、各オペレーティングシステムで部品を実行するときに適用されるロケール設定を説明します。

Windows

スクリプトまたはコマンドを対象機器で実行する場合、その対象機器のロケールと文字セットは Ops Center Automator と一致させる必要があります。ロケールと文字セットは、Windows の［**コントロール パネル**］の［**地域と言語**］領域にある、日時の形式、ユーザーレベルの表示言語、システムレベルの表示言語、およびシステムロケール設定を制御する設定によって決定されます。

Linux

部品の実行時に適用されるロケール設定は、［部品作成］ダイアログまたは［部品編集］ダイアログの［**文字セット自動判定**］設定によって異なります。

- チェックボックスがオフになっている場合、スクリプトは LC_ALL=C ロケールで実行されます。コマンドおよびコマンドパラメーターは ASCII 文字だけで構成する必要があります。コマンドパラメーター、標準出力、または標準エラー出力に非 ASCII 文字が含まれている場合、文字化けが発生し、コマンドが正常に実行できなくなる可能性があります。
- チェックボックスがオンになっている場合、スクリプトは接続ユーザーのデフォルト値のロケールを使用して実行されます。操作の対象機器でスクリプトまたはコマンドを実行する場合、環境変数 LC_ALL および LANG は接続ユーザーのデフォルト値のロケールに設定されます。その他の環境変数は変更されません。

スクリプトまたはコマンドを実行すると、次の優先順位でロケールが割り当てられます。

優先順位	環境変数
1	LC_ALL の値
2	LC_CTYPE の値
3	LANG の値

B

ビルトイン部品の説明

ここでは、Service Builder で事前に設定された部品について説明します。

B.1 汎用コマンド実行部品

汎用コマンド実行部品では、接続先ホストでコマンドラインを実行します。

［エージェントレス接続先定義］ビューで認証情報をあらかじめ設定している場合、汎用コマンド実行部品に以下の情報を指定してコマンドを実行できます。

- どのデバイスでコマンドを実行するか（destinationHost プロパティ）
- 実行するコマンド（commandLine プロパティ）
- コマンド引数（commandLineParameter プロパティ）

接続先のホストが Ops Center Automator サーバ（localhost）の場合、ユーザー ID とパスワードは不要です。

コマンドを操作対象機器で実行する場合、Ops Center Automator サーバおよび操作対象機器のオペレーティングシステムのコマンドで使用できる文字を指定します。例えば、Ops Center Automator サーバと操作対象機器でロケールが日本語に設定されている Windows が実行されている場合、MS932 文字セットの文字を指定できます。

ローカル実行機能が有効になっていて、自ホストの OS が Windows の場合、コマンドは System アカウント権限で実行されます。OS が Linux の場合、コマンドは root ユーザー権限で実行されます。コマンドからの実行フォルダは、次のように指定されます。

- 接続先で Windows が実行されている場合：Admin$¥Hitachi¥CMALib¥HAD¥home Admin$は windir 環境変数に指定されているフォルダ
- 接続先で Linux が実行されており、elevatePrivileges プロパティに true が指定されている場合：root ユーザーのホームディレクトリ
- 接続先で Linux が実行されており、elevatePrivileges プロパティに false が指定されている場合：接続ユーザーのホームディレクトリ

実行の前提条件

汎用コマンド実行部品を使用する前に、操作対象機器のオペレーティングシステムに特定のコマンドをインストールする必要があります。

操作対象機器で Windows が実行されている場合、汎用コマンド実行部品を使用するには、管理共有を有効にする必要があります。

注意事項

- Linux で Ops Center Automator が実行されており、接続先ホストが Windows の場合、IPv6 はサポートされません。
- 実行時のロケールと文字セットは、操作対象機器の OS によって異なります。
- 部品の実行中にタスクの実行が停止した場合、タスクの状態は汎用コマンド実行部品の処理が終了したときに「失敗」または「正常終了」になります。部品の実行が終了した後のステップとタスクの状態は、ステップの終了コードおよび後続ステップの実行条件によって異なります。後続ステップの実行条件（［後続ステップ実行条件］）は、［ステップ作成］ダイアログまたは［ステップ編集］ダイアログで設定できます。
- 実行方法は、操作対象機器の OS によって異なります。コマンドは、Windows では SMB および RPC によって、Linux では SSH によって実行されます。したがって、Linux ベースの操作対象機器では SSH サーバをセットアップする必要があります。

- SSH で使用するポート番号を接続先プロパティファイル（*connection-destination-name*.properties）またはプロパティファイル（config_user.properties）で設定できます。
- 操作対象機器で Windows が実行されている場合、ユーザープロファイルは引き継がれません。つまり、部品は、デスクトップで実行されたコマンドまたはスクリプトとは異なる実行結果を生成する可能性があります。この問題を回避するには、部品を実行するときに、ユーザー環境変数、レジストリエントリ、プロキシ設定など、ユーザープロファイル内の設定を参照しないようにします。ユーザープロファイルの要素を参照するコマンドやスクリプトは、予期したように動作しない場合があります。例えば、システムのプロキシ設定を参照するコマンドまたはスクリプトを実行した場合、それらは通信エラーで失敗することがあります。この問題は、スクリプトを使用して Windows Update を実行するシナリオなどで発生する可能性があります。
- 操作対象機器で Linux が実行されており、ASCII 以外の文字を commandLine プロパティまたは commandLineParameter プロパティに指定する必要がある場合、次に示す部品では、ログインスクリプトの設定が必須となります。
  - 汎用コマンド実行部品
  - ファイル転送部品
  - カスタム部品
  - ターミナルコマンド実行部品

メモ　remoteFilePath プロパティの値に ASCII 以外の文字が含まれる場合、ファイル転送部品にのみ適用されます。

ログインスクリプトの設定

接続ユーザーのログインスクリプトで istrip の設定を無効にします。
各部品のコマンドラインで **stty -a** コマンドを実行すると、コマンドの標準出力で istrip の設定状況を確認できます。「-istrip」と表示されていれば、istrip の設定は無効です。istrip の表示に「-」がない場合、istrip の設定は有効のため、接続ユーザーのログインスクリプトに **stty -istrip** コマンドを記述してください。

重要　root ユーザーに昇格する設定をしている場合は、root ユーザーのログインスクリプトで istrip の設定が上書きされるため、あらかじめ、root ユーザーのログインスクリプトで istrip の設定が無効になっていることを確認してください。

終了コード

汎用コマンド実行部品では、次の終了コードが生成されます。

終了コード	説明
0～63	コマンドまたはスクリプトの終了コード（0～63）は、部品の終了コードとして返されます。終了コードの意味は、コマンドまたはスクリプトによって異なります。
64	指定したコマンドまたはスクリプトの終了コードが 64 以上の場合、部品の終了コードとして 64 が返されます。

終了コード	説明
65	Ops Center Automator サーバとの接続に失敗しました。例えば、部品の実行時に Ops Center Automator サーバが停止した可能性があります。
66	次のユーザーが Ops Center Automator ユーザーにマッピングされています。 • Administrators group に属さないユーザー。 • UAC が有効な環境で、Administrators group に属するビルトイン Administrator 以外のユーザー。
68	対象のジョブ実行 ID についての情報がありません。
69	タスク処理エンジンの環境変数を取得できません。
70	操作対象機器との接続に失敗しました。
71	コマンドの実行に失敗しました。
72	コマンドの実行状態を取得できません。
76	接続のタイムアウトが発生しました。
77	操作対象機器のホスト名を解決できません。
78	次のいずれかの理由で操作対象機器との認証に失敗しました。 • パスワード認証に失敗した。 • 操作対象機器で公開鍵認証がセットアップされていない。 • 公開鍵の認証時に、秘密鍵がパスフレーズと一致しなかった。 • 公開鍵の認証時に、操作対象機器に登録されている公開鍵と秘密鍵が対応しなかった。 • 公開鍵の認証時に、無効な秘密鍵が使用された。
80	タスクの実行が停止しました。
81	部品が不正な状態で呼び出されました。
82	タスク処理エンジンからの要求メッセージを正しく解析できません。
83	Ops Center Automator サーバの環境が破損しています。
84	指定された部品についての情報を取得できません。
86	指定されたプロパティの値が無効です。
127	そのほかのエラーが発生しました。

プロパティリスト

汎用コマンド実行部品では、次のプロパティを使用できます。

プロパティキー	プロパティ名	説明	入出力タイプ
destinationHost※	対象ホスト	操作対象機器の IPv4 アドレス、IPv6 アドレス、またはホスト名を指定します。ホスト名は 1,024 文字以内で指定する必要があります。 Ops Center Automator サーバと操作対象機器をネットワークを使用して接続する必要があり	入力

プロパティキー	プロパティ名	説明	入出力タイプ
		ます。複数の IP アドレスまたはホスト名を指定することはできません。	
credentialType※	認証種別	コマンドまたはスクリプトの実行時に使用する認証タイプとして、次のどちらかを指定します。 **Destination** [エージェントレス接続先定義]ビューに設定されている認証情報を使用するには、このオプションを指定します。[destination]を指定すると、Ops Center Automator ログインユーザーの IP アドレスに従って、接続先定義で設定されている Windows または SSH の認証情報が適用されます。認証情報に関連するプロパティ（account、password、suPassword、publicKeyAuthentication）と keyboardInteractiveAuthentication の指定を省略することができます。 **Property** 以下のプロパティに指定された値を認証情報として使用するには、このオプションを指定します。 ・ account ・ password ・ suPassword ・ publicKeyAuthentication ・ keyboardInteractiveAuthentication	入力
account	ユーザー ID	操作対象機器へのログインに使用するユーザー ID を最大 256 文字で指定します。 ドメインユーザーは、以下のどちらの形式でも指定できます。 ・ ドメイン名¥ユーザー名 ・ ユーザー名@ドメイン名	入力
password	パスワード	操作対象機器へのログインに使用するパスワードを最大 256 文字で指定します。操作対象機器で Linux が実行されており、publicKeyAuthentication プロパティに true が指定されている場合、このプロパティを省略できます。	入力
suPassword	root のパスワード	操作対象機器の OS が Linux の場合、root パスワードを最大 256 文字で指定します。OS が Windows の場合、このプロパティを指定する必要はありません。操作対象機器で Windows が実行されている場合や、elevatePrivileges プロパティに false が指定されている場合も、このプロパティを省略できます。	入力
runAsSystem	システムアカウントで実行	Windows ホストで実行する場合、[true]を指定するとシステムアカウントを使用して、[false]を指定すると別の接続ユーザーを使用してコマンドを実行できます。このプロパティの指定値に関係なく、接続されたユーザーは、エージェントレス接続先定義または部品のプロ	入力

プロパティキー	プロパティ名	説明	入出力タイプ
		パティに指定されたユーザーになります。値は大文字小文字を区別しません。値を指定しない場合は false とみなされます。接続先ホストの OS が Windows 以外の場合、この値は無視されます。 デフォルト値は false です。	
publicKeyAuthentication	SSH 公開鍵認証設定	操作対象機器の OS が Linux の場合、公開鍵認証を使用するかどうかに応じて、以下のどちらかを指定します。値は大文字小文字を区別しません。値を指定しない場合は false とみなされます。操作対象機器で Windows が実行されている場合、このプロパティを省略できます。 **true** 公開鍵認証を使用するには、このオプションを指定します。 **false** パスワードまたはキーボードインタラクティブ認証を使用するには、このオプションを指定します。 デフォルト値は false です。	入力
keyboardInteractiveAuthentication	キーボードインタラクティブ認証設定	Linux 環境で SSH キーボードインタラクティブ認証を使用するかどうかを制御します。接続先のオペレーティングシステムが Linux の場合、システムはキーボードインタラクティブ認証を使用するかどうかを切り替えます。このプロパティを true に設定した場合、キーボードインタラクティブ認証を使用します。このプロパティを false に設定した場合、キーボードインタラクティブ認証を使用しません。このプロパティは大文字小文字を区別しません。このプロパティは、publicKeyAuthentication を false に設定している場合のみ有効です。このプロパティが存在しない（以前の部品バージョンの場合）または値が指定されていない場合、このプロパティは false とみなされます。 デフォルト値は false です。	入力
elevatePrivileges	権限昇格	操作対象機器の OS が Linux の場合、ユーザーを root 権限に昇格する必要があるかどうかに応じて、以下のどちらかを指定します。値は大文字小文字を区別しません。値を指定しない場合は true とみなされます。操作対象機器で Windows が実行されている場合、このプロパティを省略できます。 **true** root 権限を持つユーザーとしてコマンドを実行するには、このオプションを指定します。 **false** ユーザーを root に昇格することなくコマンドを実行するには、このオプションを指定します。コマンドは、接続ユーザーの権限で実行されます。 デフォルト値は false です。	入力

プロパティキー	プロパティ名	説明	入出力タイプ
commandLine※	コマンドライン	操作対象機器で実行するコマンドまたはスクリプトの絶対パスを最大 256 文字で指定します。Windows では、部品プロパティ「runAsSystem」の指定値に従って実行ユーザーが変更されます。コマンドライン内の環境変数を表す特殊文字はエスケープされません。特殊文字を文字列として処理するには、Windows ではパーセント（%）、Linux では円記号（¥）を使用して文字をエスケープします。コマンドまたはスクリプトは、次のユーザーの権限に従って実行されます。 操作対象機器で Windows が実行されている場合 ・ credentialType プロパティに［destination］を指定した場合、［エージェントレス接続先定義］ビューで設定したユーザーの権限でコマンドが実行されます。 ・ credentialType プロパティに［property］を指定した場合、account プロパティに指定したユーザーの権限でコマンドが実行されます。 操作対象機器で Linux が実行されている場合 ・ credentialType プロパティに［destination］を指定した場合、elevatePrivileges プロパティの値に応じて［エージェントレス接続先定義］ビューで設定した root ユーザーまたはユーザーの権限でコマンドが実行されます。 ・ credentialType プロパティに［property］を指定した場合、elevatePrivileges プロパティの値に応じて account プロパティに指定した root ユーザーまたはユーザーの権限でコマンドが実行されます。	入力
commandLineParameter	コマンドラインパラメーター	コマンドまたはスクリプトの引数を最大 1,024 文字で指定します。 Ops Center Automator サーバおよび操作対象機器の OS のコマンドラインに入力できる文字をコマンドラインパラメーターとして指定します。コマンドライン内の環境変数を表す特殊文字はエスケープされません。特殊文字を文字列として処理するには、Windows ではパーセント（%）、Linux では円記号（¥）を使用して文字をエスケープします。環境変数をコマンドラインパラメーターの値として指定することもできます。指定形式は、操作対象機器の OS によって異なります。 操作対象機器で Windows が実行されている場合： % 環境変数% 操作対象機器で Linux が実行されている場合： $ 環境変数	入力

プロパティキー	プロパティ名	説明	入出力タイプ
outputCondition	出力条件	標準出力プロパティ 1～3 に出力する条件を指定します。次の値を指定できます。 ・ always -- 指定されたパターンと一致しなくても、null 文字を出力します。 ・ patternMatch -- 標準出力パターン 1～3 に一致する場合だけ出力します。 出力プロパティに出力がない場合、マッピングされたサービスプロパティも更新されません。デフォルト値は always です。	入力
stdoutProperty1	標準出力プロパティ 1	stdoutPattern1 プロパティから抽出された文字列は、このプロパティに出力されます。	出力
stdoutPattern1	標準出力パターン 1	stdoutProperty1 プロパティに出力する標準出力の正規表現パターンを最大 1,024 文字で指定します。正規表現パターンは PCRE に準拠する形式で指定します。stdoutProperty1 プロパティにサービスプロパティのキーを指定し、stdoutPattern1 プロパティを指定しない場合、commandLine プロパティに指定したコマンドまたはスクリプトの標準出力と標準エラー出力全体がサービスプロパティに割り当てられます。	入力
stdoutProperty2	標準出力プロパティ 2	stdoutPattern2 プロパティから抽出された文字列は、このプロパティに出力されます。	出力
stdoutPattern2	標準出力パターン 2	標準出力から stdoutProperty2 プロパティへ出力される正規表現パターンを、最大 1,024 文字で指定します。正規表現パターンは PCRE に準拠する形式で指定します。stdoutProperty2 プロパティにサービスプロパティのキーを指定し、stdoutPattern2 プロパティを指定しない場合、commandLine プロパティに指定したコマンドまたはスクリプトの標準出力と標準エラー出力全体がサービスプロパティに割り当てられます。	入力
stdoutProperty3	標準出力プロパティ 3	stdoutPattern3 プロパティから抽出された文字列は、このプロパティに出力されます。	出力
stdoutPattern3	標準出力パターン 3	stdoutProperty3 プロパティに出力する標準出力の正規表現パターンを最大 1,024 文字で指定します。正規表現パターンは PCRE に準拠する形式で指定します。stdoutProperty3 プロパティにサービスプロパティのキーを指定し、stdoutPattern3 プロパティを指定しない場合、commandLine プロパティに指定したコマンドまたはスクリプトの標準出力と標準エラー出力全体がサービスプロパティに割り当てられます。	入力

注※　必須プロパティです。

これらのプロパティに指定したコマンドおよびスクリプトの標準出力、または標準エラー出力は、Ops Center Automator 内のステップの標準出力として出力されます。ただし、コマンドまたはス

クリプトの標準出力と標準エラー出力の合計が 100 KB を超える処理は、製品サポートの範囲外です。コマンドまたはスクリプトを事前に実行し、標準出力と標準エラー出力の合計が 100 KB を超えないことを確認してください。

操作対象機器で Windows が実行されている場合、commandLine プロパティと commandLineParameter プロパティに指定した内容でバッチファイルが作成され、その内容が操作対象機器で実行されます。したがって、この操作の結果は、コマンドプロンプトから同じコマンドとスクリプトを実行した場合の結果とは異なる可能性があります。

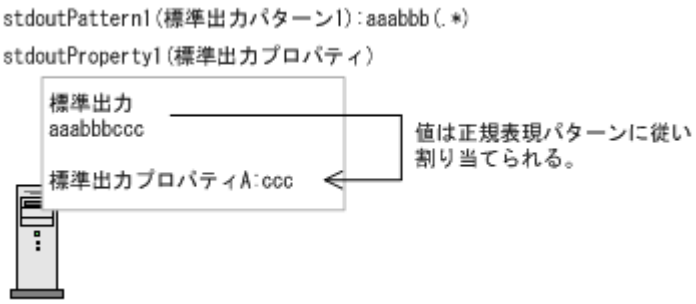
操作対象機器で Linux が実行されている場合、標準出力および標準エラー出力内の行送りコードは次のように変更されます。

- CR（0x0d）は LF（0x0a）に変更されます
- CR+LF（0x0d0a）は LF+LF（0x0a0a）に変更されます。

また、標準出力および標準エラー出力の末尾の文字が行送りコード（CR、LF、または CR+LF）ではない場合、末尾に LF（0x0a）が追加されます。

stdoutPattern プロパティと stdoutProperty プロパティの使用例

stdoutPattern プロパティを使用して、標準出力に出力する値を抽出し、stdoutProperty プロパティに格納することができます。以下の図は、stdoutPattern1 に「aaabbb(.*)」を指定した場合のデータフローを示しています。



stdoutPattern1 で定義されているように、標準出力「aaabbbccc」では、「aaabbb」以降の値（この場合は「ccc」）が抽出されます。抽出された値は stdoutProperty1 プロパティに格納されます。

SSH ポート番号を指定する

SSH を使用して操作対象機器に接続する場合、ポート番号を指定できます。以下の表で、ポート番号の指定方法と各方法の優先順位について説明します。

優先順位	設定先	プロパティキー	デフォルト値
1	接続先プロパティファイル（*connection-destination-name*.properties）	ssh.port	--
2	プロパティファイル（config_user.properties）	ssh.port.number	22

B.2 ファイル転送部品

ファイル転送部品では、ファイルやフォルダを相手ホストとの間で転送します。

[エージェントレス接続先定義] ビューで認証情報をあらかじめ設定している場合、以下の情報を指定してファイル転送部品を実行できます。

- 操作対象機器（remoteHost プロパティ）
- 転送モード（transferMode プロパティ）
- Ops Center Automator サーバのファイルまたはフォルダのパス（localFilePath プロパティ）
- 操作対象機器のファイルまたはフォルダのパス（`remoteFilePath` プロパティ）

エージェントレス接続先への転送用のファイルパスには、Ops Center Automator サーバおよび操作対象機器のオペレーティングシステムのコマンドで使用できる文字を指定します。例えば、Ops Center Automator サーバと操作対象機器の両方が日本語バージョンの Windows を実行している場合、MS932 文字セットの文字を指定できます。

接続先のホストが Ops Center Automator サーバ（`localhost`）の場合、ユーザー ID とパスワードは不要です。

操作対象機器で Windows が実行されている場合、認証情報に設定されたユーザーがファイルを転送します。

操作対象機器で Linux が実行されている場合、`elevatePrivileges` プロパティの値に応じて root ユーザーまたは接続ユーザーの権限に従ってファイルが転送されます。

メモ　ローカル実行機能が有効になっている場合、ファイルは転送されません。自ホストの OS が Windows の場合、System アカウント権限でファイルが自ホストにコピーされます。自ホストの OS が Linux の場合、root ユーザー権限でファイルが自ホストにコピーされます。

前提条件

操作対象機器の OS に応じて、以下のように環境を構成します。

Windows :

- Ops Center Automator サーバと操作対象機器が適切なポートを使用して通信できることを確認します。
- ファイル転送部品を実行する前に、操作対象機器で管理共有を有効にします。

Linux :

- SSH で使用するポート番号を接続先プロパティファイル（*connection-destination-name*`.properties`）またはプロパティファイル（`config_user.properties`）で設定できます。
- SCP または SFTP をサポートする SSH サーバを操作対象機器にインストールします。

ファイル転送部品を使用する前に、操作対象機器のオペレーティングシステムに特定のコマンドをインストールする必要があります。

注意事項

- Linux で Ops Center Automator が実行されており、接続先ホストが Windows の場合、IPv6 はサポートされません。
- 実行方法は、操作対象機器の OS によって異なります。ファイル転送は、Windows では RPC と CIFS（SMB）によって実行され、Linux では SSH（SCP または SFTP）によって実行されます。エージェントレス接続先の定義でプロトコルを選択する場合、Windows では Windows を、Linux では SSH を選択します。なお、SFTP を使用したファイル転送でのファイル転送の中断、再開機能には対応しません。
- 転送するすべてのファイルの最大合計サイズは 4GB です。
- 一度に転送できるファイルとフォルダの最大数は 10,000 です。
- 受信したファイルと同じ名前のファイルがローカルに存在する場合、システムはそのファイルの上書きを試みる場合があります。ただし、上書きされるファイルの属性が読み取り専用、隠しファイル、またはシステムファイルの場合、そのファイルを上書きできないため、転送が失敗します。
- ファイルの転送元または転送先として、Windows UNC パスまたはネットワークドライブを指定できません。
- Ops Center Automator がインストールされているマシンと接続先のホストでは、ファイルとフォルダ自体で必要とされる空き領域に加えて、転送ファイルの 2 倍のサイズに相当する空き領域が一時作業エリアとして必要になります。一時作業エリアは以下のとおりです。
  - Ops Center Automator がインストールされているマシン（非クラスタ環境）：Ops Center Automator がインストールされているドライブ
  - Ops Center Automator がインストールされているマシン（クラスタ環境）：共用ディスク
  - 接続先で Windows が実行されている場合：システムドライブ
  - 接続先で Linux が実行されている場合：プロパティファイル（config_user.properties）の plugin.remoteCommand.workDirectory.ssh キーに指定されたフォルダ
- オペレーティングシステムの制限は、Ops Center Automator システムに設定されている制限を上書きします。この制限の例として、最大ファイルサイズ、フォルダごとのファイル数、ファイルとフォルダの名前の長さ、ユーザーが使用できるリソースがあります。オペレーティングシステムの制限を超えるファイル転送は製品サポートの範囲外です。Ops Center Automator サーバのオペレーティングシステムと操作対象機器のオペレーティングシステムには、Ops Center Automator の操作に影響を与えるオペレーティングシステムの制限があります。ユーザーがどのリソースを使用できるかを規定する OS の制限が、接続ユーザーおよび root 権限を持つユーザーに対して設定されています。root 権限を持つユーザーに対する制限は Linux にのみ適用されます。
- Linux を実行しているホストのフォルダをファイルの転送先として指定したときに、フォルダ内のファイルの合計サイズが 1 ファイルの最大許容サイズを超えた場合、プロセスが失敗することがあります。1 ファイルの最大サイズは、ユーザーが使用できるリソースに適用されている、ファイルシステムの制限と OS の制限によって規定されます。Ops Center Automator はファイルをアーカイブしてから送信します。つまり、アーカイブ内の個々のファイルが最大サイズより小さくても、接続先ホストの制限を超過する可能性があります。このシナリオでは、送信するフォルダ内のファイルの合計サイズを縮小するか、接続先の制限を増加します。
- 部品の実行中にタスクの実行が停止した場合、タスクの状態はファイル転送部品の処理が終了したときに「失敗」または「正常終了」になります。部品の実行が終了した後のステップとタスクの状態は、ステップの終了コードおよび後続ステップの実行条件によって異なります。後続ステップ実行条件は、［ステップ作成］ダイアログまたは［ステップ編集］ダイアログで設定できます。

終了コード

ファイル転送部品では、次の終了コードが生成されます。

終了コード	説明
0	正常に終了しました。
65	Ops Center Automator サーバとの接続に失敗しました。例えば、部品の実行時に Ops Center Automator サーバが停止した可能性があります。
66	次のユーザーが Ops Center Automator ユーザーにマッピングされています。 • Administrators group に属さないユーザー。 • UAC が有効な環境で、Administrators group に属するビルトイン Administrator 以外のユーザー。
68	対象のジョブ実行 ID についての情報がありません。
69	タスク処理エンジンの環境変数を取得できません。
70	操作対象機器との接続に失敗しました。
71	操作対象機器でのコマンドの呼び出しに失敗しました。
72	コマンドの実行状態を取得できません。
73	ファイルまたはフォルダを転送できません。
76	接続のタイムアウトが発生しました。
77	操作対象機器のホスト名を解決できません。
78	次のいずれかの理由で操作対象機器との認証に失敗しました。 • パスワード認証に失敗した。 • 操作対象機器で公開鍵認証がセットアップされていない。 • 公開鍵の認証時に、秘密鍵がパスフレーズと一致しなかった。 • 公開鍵の認証時に、操作対象機器に登録されている公開鍵と秘密鍵が対応しなかった。 • 公開鍵の認証時に、無効な秘密鍵が使用された。
80	タスクの実行が停止しました。
81	部品が不正な状態で呼び出されました。
82	タスク処理エンジンからの要求メッセージを正しく解析できません。
83	Ops Center Automator サーバの環境が破損しています。
84	指定された部品についての情報を取得できません。
86	指定されたプロパティの値が無効です。
127	そのほかのエラーが発生しました。

プロパティリスト

ファイル転送部品では、次のプロパティを使用できます。

プロパティキー	プロパティ名	説明	入出力タイプ
remoteHost※	リモートホスト	操作対象機器の IPv4 アドレス、IPv6 アドレス、またはホスト名を指定します。ホスト名は 1,024 文字以内で指定する必要があります。Ops Center Automator サーバと操作対象機器をネットワークを使用して接続する必要があります。複数の IP アドレスまたはホスト名を指定することはできません。	入力
credentialType※	認証種別	コマンドまたはスクリプトの実行時に使用する認証タイプとして、次のどちらかを指定します。 **Destination** [エージェントレス接続先定義] ビューに設定されている認証情報を使用するには、このオプションを指定します。[destination] を指定すると、Ops Center Automator ログインユーザーの IP アドレスに従って、接続先定義で設定されている Windows または SSH の認証情報が適用されます。認証情報に関連するプロパティ（account、password、suPassword、publicKeyAuthentication）と keyboardInteractiveAuthentication の指定を省略することができます。 **Property** 以下のプロパティに指定された値を認証情報として使用するには、このオプションを指定します。 ・ account ・ password ・ suPassword ・ publicKeyAuthentication ・ keyboardInteractiveAuthentication デフォルト値は destination です。	入力
account	ユーザー ID	操作対象機器へのログインに使用するユーザー ID を最大 256 文字で指定します。 ドメインユーザーは、以下のどちらの形式でも指定できます。 ・ ドメイン名¥ユーザー名 ・ ユーザー名@ドメイン名	入力
password	パスワード	操作対象機器へのログインに使用するパスワードを最大 256 文字で指定します。操作対象機器で Linux が実行されており、publicKeyAuthentication プロパティに true が指定されている場合、このプロパティを省略できます。	入力
suPassword	root のパスワード	操作対象機器の OS が Linux の場合、root パスワードを最大 256 文字で指定します。OS が Windows の場合、このプロパティを指定する必要はありません。操作対象機器で Windows が実行されている場合や、elevatePrivileges プロ	入力

プロパティキー	プロパティ名	説明	入出力タイプ
		パティに false が指定されている場合も、このプロパティを省略できます。	
publicKeyAuthentication	SSH 公開鍵認証設定	操作対象機器の OS が Linux の場合、公開鍵認証を使用するかどうかに応じて、以下のどちらかを指定します。値は大文字小文字を区別しません。値を指定しない場合は false とみなされます。操作対象機器で Windows が実行されている場合、このプロパティを省略できます。 **true** 公開鍵認証を使用するには、このオプションを指定します。 **false** パスワードまたはキーボードインタラクティブ認証を使用するには、このオプションを指定します。 デフォルト値は false です。	入力
keyboardInteractiveAuthentication	キーボードインタラクティブ認証設定	Linux 環境で SSH キーボードインタラクティブ認証を使用するかどうかを制御します。接続先のオペレーティングシステムが Linux の場合、システムはキーボードインタラクティブ認証を使用するかどうかを切り替えます。このプロパティを true に設定した場合、キーボードインタラクティブ認証を使用します。このプロパティを false に設定した場合、キーボードインタラクティブ認証を使用しません。このプロパティは大文字小文字を区別しません。このプロパティは、publicKeyAuthentication を false に設定している場合のみ有効です。このプロパティが存在しない（以前の部品バージョンの場合）または値が指定されていない場合、このプロパティは false とみなされます。 デフォルト値は false です。	入力
elevatePrivileges	権限昇格	操作対象機器の OS が Linux の場合、ユーザーを root 権限に昇格する必要があるかどうかに応じて、以下のどちらかを指定します。値は大文字小文字を区別しません。値を指定しない場合は true とみなされます。操作対象機器で Windows が実行されている場合、このプロパティを省略できます。 **true** root 権限を持つユーザーとしてコマンドを実行するには、このオプションを指定します。 **false** ユーザーを root に昇格することなくコマンドを実行するには、このオプションを指定します。コマンドは、接続ユーザーの権限で実行されます。 デフォルト値は false です。	入力
transferMode※	転送モード	転送モードとして以下のどちらかを指定します。 **send**	入力

プロパティキー	プロパティ名	説明	入出力タイプ
		Ops Center Automator サーバから操作対象機器にファイルまたはフォルダを転送する場合、このオプションを指定します。localFilePath プロパティにファイルパスを指定する場合、同じパスを remoteFilePath プロパティに指定する必要があります。1 つのファイルを転送するときに、localFilePath プロパティと remoteFilePath プロパティに異なるファイル名を指定した場合、remoteFilePath プロパティに指定したファイル名が適用されます。 **receive** 操作対象機器から Ops Center Automator サーバにファイルまたはフォルダを転送する場合、このオプションを指定します。 remoteFilePath プロパティにファイルパスを指定した場合、同じパスを localFilePath プロパティに指定する必要があります。1 つのファイルを転送するときに、remoteFilePath プロパティと localFilePath プロパティに異なるファイル名を指定した場合、localFilePath プロパティに指定したファイル名が適用されます。 デフォルト値は send です。	
localFilePath※	ローカルファイルパス	Ops Center Automator サーバ上のファイルまたはフォルダの絶対パスを 256 文字以内で指定します。localFilePath プロパティには、Ops Center Automator サーバおよび操作対象機器のオペレーティングシステムのコマンドで使用できる文字を指定します。同じ名前のファイルまたはフォルダが接続先フォルダに存在する場合、そのファイルまたはフォルダは上書きされます。このため、一意の名前を指定する必要があります。転送先のフォルダが存在しない場合、指定した構成でフォルダが作成されます。	入力
remoteFilePath※	リモートファイルパス	操作対象ホストのファイルまたはフォルダの絶対パスを 256 文字以内で指定します。 remoteFilePath プロパティには、Ops Center Automator サーバおよび操作対象機器のオペレーティングシステムのコマンドで使用できる文字を指定します。操作対象機器の OS が Linux の場合、接続ユーザーが使用している文字セットと同じ文字セットで、転送するファイルとフォルダの名前がエンコードされていることを確認します。同じ名前のファイルまたはフォルダが接続先フォルダに存在する場合、そのファイルまたはフォルダは上書きされます。このため、一意の名前を指定する必要があります。転送先のフォルダが存在しない場合、指定した構成でフォルダが作成されます。	入力

注※　必須プロパティです。

ファイルパスを指定する場合、Ops Center Automator サーバおよび操作対象機器のオペレーティングシステムのコマンドで使用できる文字を使用します。localFilePath プロパティにファイル名を指定した場合、remoteFilePath プロパティにもファイル名を指定します。localFilePath プロパティにフォルダ名を指定した場合、remoteFilePath プロパティにもフォルダ名を指定します。

localFilePath と remoteFilePath プロパティに指定できるファイルおよびフォルダにはいくつかの制限が適用されます。

操作対象機器で Windows が実行されており、Windows ファイル属性「内容を暗号化してデータをセキュリティで保護する」を持つファイルが転送ファイルの中に含まれる場合、そのファイルの転送が失敗し、部品の処理でエラーが発生します。

転送するファイルとフォルダの名前に関する制限

接続先が Windows または Linux の場合に、転送するファイルとフォルダの名前に適用される制限を以下の表に示します。

表 39 送信

ファイルまたはフォルダ	Automator 側または接続先ホスト側	プロパティ	制限
ファイル	Automator	localFilePath	ファイル名には最大 127 文字を使用できます。
	接続先ホスト	remoteFilePath	ファイル名には最大 127 文字を使用できます。
フォルダ	Automator	localFilePath	転送するフォルダ内のファイルまたはフォルダの絶対パスには、最長で 256 文字を含めることができます。 転送するフォルダから、そのフォルダの下にあるファイルまたはフォルダまでのパスの長さは最長で 127 文字とする必要があります。
	接続先ホスト	remoteFilePath	転送するフォルダ内のファイルまたはフォルダの絶対パスには、最長で 256 文字を含めることができます。 転送するフォルダから、そのフォルダの下にあるファイルまたはフォルダまでのパスの長さは最長で 127 文字とする必要があります。

表 40 受信

ファイルまたはフォルダ	Automator 側または接続先ホスト側	プロパティ	制限
ファイル	Automator	localFilePath	ファイル名には最大 127 文字を使用できます。
	接続先ホスト	remoteFilePath	ファイル名には最大 127 文字を使用できます。
フォルダ	Automator	localFilePath	転送するフォルダ内のファイルまたはフォルダの絶対パスには、最長で 256 文字を含めることができます。 転送するフォルダから、そのフォルダの下にあるファイルまたはフォルダまでのパスの長さは最長で 127 文字とする必要があります。

ファイルまたはフォルダ	Automator 側または接続先ホスト側	プロパティ	制限
	接続先ホスト	remoteFilePath	転送するフォルダ内のファイルまたはフォルダの絶対パスには、最長で 256 文字を含めることができます。 転送するフォルダから、そのフォルダの下にあるファイルまたはフォルダまでのパスの長さは最長で 127 文字とする必要があります。

SSH ポート番号を指定する

SSH を使用して操作対象機器に接続する場合、ポート番号を指定できます。以下の表で、ポート番号の指定方法と各方法の優先順位について説明します。

優先順位	設定先	プロパティキー	デフォルト値
1	接続先プロパティファイル (*connection-destination-name*.properties)	ssh.port	--
2	プロパティファイル (config_user.properties)	ssh.port.number	22

SSH ファイル転送プロトコルを指定する

SSH を使用して操作対象機器に接続する場合、SFTP を使用するかどうかを指定できます。SFTP を使用しない場合は SCP を使用します。以下の表で、SFTP の指定方法と各方法の優先順位について説明します。

優先順位	設定先	プロパティキー	デフォルト値
1	接続先プロパティファイル (*connection-destination-name*.properties)	sftp.enable	--
2	プロパティファイル (config_user.properties)	plugin.sftp.enable	false

転送されたファイルを処理する

転送されたファイルは、操作対象機器の OS および transferMode プロパティに指定された値に応じて異なる方法で処理されます。以下の表で、転送されたファイルの処理方法について説明します。

表 41 Windows

項目			送信	受信
転送されたファイルのタイムスタンプ	ファイルの作成時	作成日時	転送日時	転送日時
		更新日時	転送元ファイルの更新日時	転送元ファイルの更新日時
		アクセス日時	転送日時	転送日時

項目			送信	受信
	ファイルの上書き時	作成日時	上書きされたファイルの作成日時	上書きされたファイルの作成日時
		更新日時	転送元ファイルの更新日時	転送元ファイルの更新日時
		アクセス日時	上書きされたファイルのアクセス日時	上書きされたファイルのアクセス日時
転送元ファイルに対して必要とされるアクセス権限			システムアカウントの読み取り権限	システムアカウントの読み取り権限
転送先ファイルの親フォルダに対して必要とされるアクセス権限			認証情報に設定されたユーザーの書き込み権限	システムアカウントの書き込み権限
転送先ファイルを上書きするときにそのファイルで必要とされるアクセス権限			認証情報に設定されたユーザーの書き込み権限	システムアカウントの書き込み権限
転送先ファイルに割り当てられたアクセス権限	ファイルの作成時		親フォルダの権限を引き継ぐ	親フォルダの権限を引き継ぐ
	ファイルの上書き時		上書きされたファイルの権限を引き継ぐ	上書きされたファイルの権限を引き継ぐ

表 42 Linux

項目			送信	受信
転送されたファイルのタイムスタンプ	ファイルの作成時	作成日時	転送日時	転送日時
		更新日時	転送日時	転送日時
		アクセス日時	転送日時	転送日時
	ファイルの上書き時	作成日時	転送日時	上書きされたファイルの作成日時
		更新日時	転送日時	転送日時
		アクセス日時	上書きされたファイルのアクセス日時	上書きされたファイルのアクセス日時
転送元ファイルに対して必要とされるアクセス権限			システムアカウントの読み取り権限	接続ユーザーの読み取り権限
転送先ファイルの親フォルダに対して必要とされるアクセス権限			接続ユーザーの書き込み権限	システムアカウントの書き込み権限
転送先ファイルを上書きするときにそのファイルで必要とされるアクセス権限			接続ユーザーの書き込み権限	システムアカウントの書き込み権限
転送先ファイルに割り当てられたアクセス権限	ファイルの作成時		root または接続ユーザーの umask 値を使用する	親フォルダの権限を引き継ぐ
	ファイルの上書き時		上書きされたファイルの権限を引き継ぐ	上書きされたファイルの権限を引き継ぐ

B.3 繰り返し実行部品

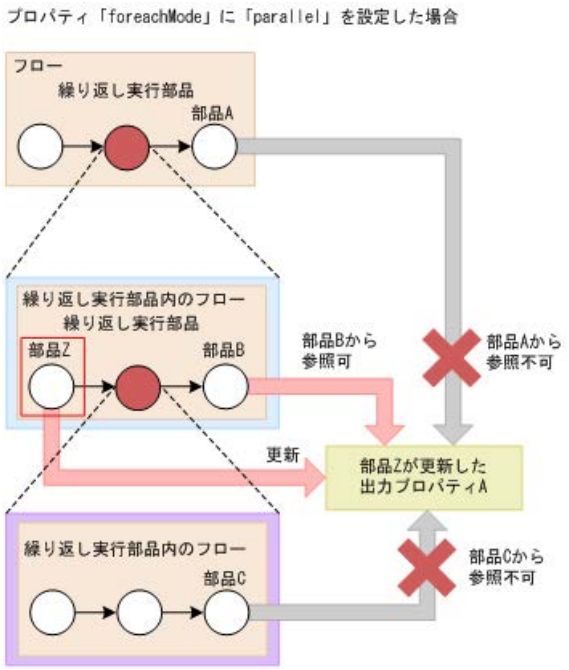
繰り返し実行部品は、特定のフローを繰り返し実行します。

フローの繰り返しごとに、入力プロパティ（inputProperties）に値を指定してサービスを実行できます。これは例えば、異なるサーバに同じ処理を実行したい場合などに便利です。フローの実行方法には、並行して複数のフローを実行する同時実行、および現在のフローの実行が終了したら次のフローを実行する順次実行があります。また、繰り返し実行部品はネストされた構成にすることができ、3 レベルまでの繰り返し実行部品のネストを定義することが可能です。階層フロー部品と組み合わせて使用する場合、あわせて 25 レベルまでまで定義することが出来ます。

注意事項

- **foreachMode** プロパティで［parallel］オプションが指定されている場合、繰り返しのタスク内で、参照または更新されるプロパティ（サービスプロパティ、部品の出力プロパティ、変数）の値は、同じ繰り返しのタスク（n 番目のフロー）でのみ有効です。加えて、プロパティ（サービスプロパティ、部品の出力プロパティ、変数）の値は、同時に処理される繰り返しのタスク（n 番目のフローを除く）、あるいは下位レベルでの繰り返しのタスクでは共有できません。

次の図は、各レベルの部品が、部品 Z によって更新された出力プロパティをどのように参照するかを示します。



フロー数の制約条件

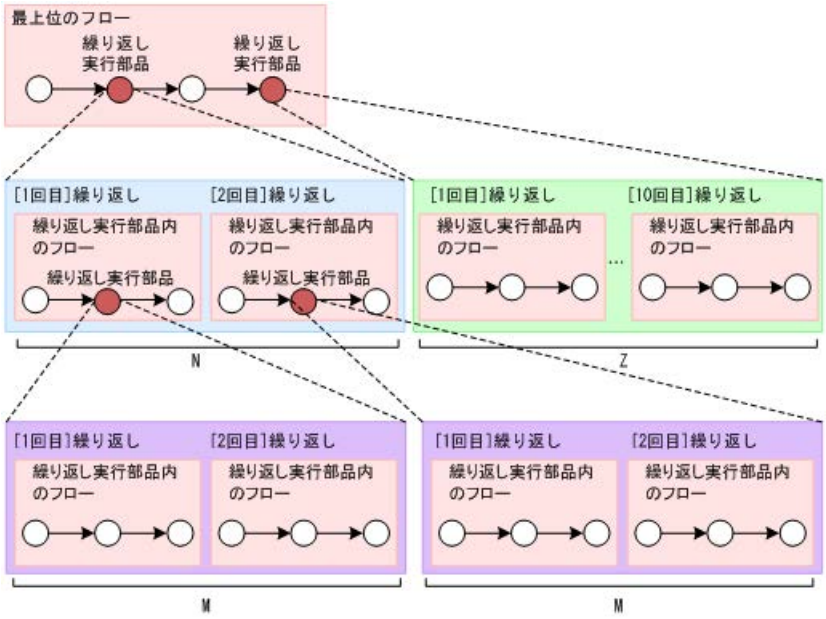
繰り返し実行部品に指定できるフローの最大数は、**inputProperties** プロパティを通して設定され、各サービスごとに 10,000 を超えることは出来ません。フローの総数は、最上位レベルのフローで実行されたフローの数、または階層フロー部品と関連づけられたフローの数は含みません。

サービスにおいて繰り返し中のフローの数が増えると、応答が遅れたり、ブラウザがクラッシュする原因にさえなることがあります。そのため、サービスにおいて繰り返し中のフロー数には上限が設定されます。

この数に含まれるのは繰り返し実行部品のフローのみになります。階層フロー部品のフローはカウントされません。

種別	カウント対象
繰り返し実行部品下のフロー	Yes
階層フロー部品によるフロー	Yes
他のフロー	No

次の例では、繰り返し実行部品の **inputProperties** プロパティが、部品 N を 2 回、部品 M を 2 回、部品 Z を合計 10 回繰り返すよう設定されています。よって、フロー数の合計は以下のように計算できます。N+N*M+Z (2+2*2+10) = 16.



次の予約プロパティを使用して、ネストされた構成における、繰り返し実行部品の入力値とループインデックス値を指定できます。

- reserved.loop.inputN
- reserved.loop.indexN

メモ　繰り返し実行部品がネストされている場合、デバッグ中にブラウザに障害が出る可能性があります。この場合、各繰り返し実行部品の"inputProperties"プロパティに設定されている値を小さくする必要があります。その後デバッグを繰り返してください。

メモ　フローの合計数が 10,000 を超えないように、各繰り返し実行部品に"inputProperties"プロパティを設定してください。階層フロー部品のフローは、フロー数の計算には含まれません。

終了コード

繰り返し実行部品では、次の終了コードが生成されます。

終了コード	説明
0	正常に終了しました。

終了コード	説明
1	繰り返し処理の一部が失敗しました。
2	繰り返し処理すべてが失敗しました。
3	サービスにおける、繰り返し実行部品下のフローの合計数が、上限を超えています。
65	Ops Center Automator サーバとの接続に失敗しました。例えば、部品の実行時に Ops Center Automator サーバが停止した可能性があります。
66	次のユーザーが Ops Center Automator ユーザーにマッピングされています。 ・ Administrators group に属さないユーザー。 ・ UAC が有効な環境で、Administrators group に属するビルトイン Administrator 以外のユーザー。
68	対象のジョブ実行 ID についての情報がありません。
69	タスク処理エンジンの環境変数を取得できません。
80	タスクの実行が停止しました。
81	部品が不正な状態で呼び出されました。
82	タスク処理エンジンからの要求メッセージを正しく解析できません。
83	Ops Center Automator サーバの環境が破損しています。
84	指定された部品についての情報を取得できません。
86	指定されたプロパティの値が無効です。
127	そのほかのエラーが発生しました。

プロパティリスト

繰り返し実行部品のプロパティは次のとおりです。

プロパティキー	プロパティ名	説明	入出力タイプ
inputProperties※	繰り返し入力プロパティ	フローの繰り返しごとに、1,024 文字以内で入力プロパティの値を指定します。繰り返しごとに異なるプロパティを指定できます。プロパティを区切るにはコンマを使用します。コンマは区切り文字としてのみ使用できます。最大の繰り返し数は 99 回です。 100 個以上のコンマ区切りの値を指定することはできません。	入力
outputProperties	繰り返し出力プロパティ	繰り返しの回数、出力プロパティの値を出力します。合計出力は 1,024 文字以内です。各繰り返しで、1 つのプロパティ値が inputProperties プロパティで指定された順番でコンマで区切られて出力されます。区切り文字としてコンマを使用します。	出力
outputResult	繰り返し実行結果	各フローの実行結果がコンマ区切りで出力されます。 **true** フローの実行が成功した場合に出力されます。	出力

プロパティキー	プロパティ名	説明	入出力タイプ
		false フローの実行が失敗した場合に出力されます。	
foreachMode※	繰り返し実行方式	繰り返しのフローに実行方法を指定します。 **parallel** 繰り返しのフローが並行して実行されます。最大 99 個のフローを並行して実行できます。最大数を超過すると、実行中のフローの数が最大数より少なくなったときに残りのフローが実行されます。プロパティファイル（config_user.properties）で foreach.max_value キーを使用すると、同時に実行できるフローの数を 1～99 の間で変更できます。エラーが発生した場合でも、未実行のすべてのフローが実行されます。 **serial** 繰り返しのフローが順次実行されます。エラーが発生した場合、未実行のフローは実行されません。 デフォルト値は parallel です。	入力

注※　必須プロパティです。

B.4 メール通知部品

メール通知部品を使用して、指定した宛先にメールを送信します。

この部品では、SMTP サーバに接続し、受信者、件名、および本文を指定してメールを送信できます。

実行の前提条件

ビルトインのサービス共有プロパティから以下の情報が取得されます。したがって、［システム設定］ビューでこれらの項目の値を事前に設定しておきます。

- SMTP サーバのアドレス
- ポート番号
- ユーザー ID
- パスワード
- 通知メールの送信元

注意事項

- toAddress、ccAddress、および bccAddress プロパティを指定していない場合も、終了コードは 0 になります。
- 指定するメールアドレスは、ビルトインのサービス共有プロパティの値と異なります。したがって、toAddress、ccAddress、bccAddress プロパティの少なくとも 1 つを必ず指定します。
- toAddress、ccAddress、bccAddress プロパティのいずれかに無効なメールアドレスを指定した場合、すべてのアドレスへのメールの送信が失敗します。

- mailSubject または mailBody プロパティ内で機種依存文字を使用している場合や、文字セット間で互換性のない文字を使用している場合、その文字は疑問符（?）または他の文字で置き換えられます。この場合は、メールの文字を変更するか、エンコードを変更します。
- 次の文字は正しく変換されない可能性があります。

  ~, ¥, ＼, ～, ∥, －, ¢, £, ¬

- 部品の実行中にタスクの実行が停止した場合、タスクの状態はメール通知部品の処理が終了したときに「失敗」または「正常終了」になります。部品の実行が終了した後のステップとタスクの状態は、ステップの終了コードおよび後続ステップの実行条件によって異なります。後続ステップの実行条件（[後続ステップ実行条件]）は、[ステップ作成] ダイアログまたは [ステップ編集] ダイアログで設定できます。

終了コード

メール通知部品では、次の終了コードが生成されます。

終了コード	説明
0	正常に終了しました。
65	Ops Center Automator サーバとの接続に失敗しました。例えば、部品の実行時に Ops Center Automator サーバが停止した可能性があります。
66	次のユーザーが Ops Center Automator ユーザーにマッピングされています。 • Administrators group に属さないユーザー。 • UAC が有効な環境で、Administrators group に属するビルトイン Administrator 以外のユーザー。
68	対象のジョブ実行 ID についての情報がありません。
69	タスク処理エンジンの環境変数を取得できません。
70	SMTP サーバとの接続に失敗しました。
78	認証が失敗しました。
79	メールの送信に失敗しました。
80	タスクの実行が停止しました。
81	部品が不正な状態で呼び出されました。
82	タスク処理エンジンからの要求メッセージを正しく解析できません。
83	Ops Center Automator サーバの環境が破損しています。
84	指定された部品についての情報を取得できません。
86	指定されたプロパティの値が無効です。
127	そのほかのエラーが発生しました。

プロパティリスト

メール通知部品では、次のプロパティを使用できます。

プロパティキー	プロパティ名	説明	デフォルト値	入出力タイプ	必須
toAddress	TO メールアドレス	受信者のメールアドレスを 1,024 文字以内で指定し、TO 属性に入力します。複数のアドレスを指定する場合、各アドレスをコンマで区切ります。	--	入力	任意
ccAddress	CC メールアドレス	受信者のメールアドレスを 1,024 文字以内で指定し、CC 属性に入力します。複数のアドレスを指定する場合、各アドレスをコンマで区切ります。	--	入力	任意
bccAddress	BCC メールアドレス	受信者のメールアドレスを 1,024 文字以内で指定し、BCC 属性に入力します。複数のアドレスを指定する場合、各アドレスをコンマで区切ります。	--	入力	任意
encodeType	エンコード種別	次のうちからメールのエンコードを 1 つ指定します。 • us-ascii • iso-2022-jp • shift_jis • euc-jp • utf-8	utf-8	入力	必須
mailSubject	メール件名	メールの件名行を 256 文字以内で指定します。	--	入力	必須
mailBody	メール本文	メールの本文テキストを 1,024 文字以内で指定します。	--	入力	任意

B.5 ユーザー応答待ち部品

ユーザー応答待ち部品を使用すると、オペレーターはサービスの実行中に次のステップの処理を選択できます。

応答可能なオペレーターは、すべてのユーザーを対象とすることも、特定のユーザーに限定することも可能です。

オペレーターは［応答入力］ダイアログを使用して処理を選択します。また、タスクが応答待ちであることをオペレーターへ通知する電子メールを設定することもできます。

［応答入力］ダイアログには、次の方法でアクセスします。

- 応答待機通知メールに記載されている URL からのリンク
- ［タスク］ビューからのリンク

実行の前提条件

ビルトインのサービス共有プロパティから以下の情報が取得されます。このため、タスクが応答待ちのときオペレーターへ通知するには、［システム設定］ビューから事前にこれらの項目の値を設定します。

- SMTP サーバのアドレス
- ポート番号
- ユーザー ID
- パスワード
- 通知メールの送信元

注意事項

- 次のいずれかの条件が当てはまる場合、タスクがユーザー応答待ちであるという電子メールの通知は送信されません。
  - ビルトインのサービス共有プロパティに値が設定されていない。
  - SMTP がセットアップされていない。
  - toAddress、ccAddress、および bccAddress プロパティが、どれも指定されていない。
  - toAddress、ccAddress、および bccAddress プロパティのどれかに、無効な電子メールアドレスが指定されている。
- 指定するメールアドレスは、ビルトインのサービス共有プロパティの値と異なります。したがって、toAddress、ccAddress、bccAddress プロパティの少なくとも 1 つを必ず指定します。
- ［応答入力］ダイアログが表示され、オペレーターの応答待ちの間は、ユーザー応答待ち部品の実行を停止しないでください。実行を停止すると、オペレーターが次のステップの処理を選択していてもエラーが発生します。
- 通知電子メールの本文には、［応答入力］ダイアログへリンクする URL が自動的に入力されます。特定のタスクについて、複数のステップが応答待ちの場合、ユーザー応答待ち部品を実行する各ステップに別の URL が割り当てられ、そのステップ用の［応答入力］ダイアログにそれぞれの URL が表示されます。
- ［応答入力］ダイアログへリンクする URL 以外で応答入力する際、複数の応答待ちが存在する場合は、最も古い応答待ちから応答入力する必要があります。
- ［応答入力］ダイアログのレイアウトは変更できません。
- labelButton1 および labelButton9 プロパティからの終了コードは異常終了とみなされ、エラー情報がタスクログへ出力されます。labelButton0 プロパティと、labelButton1 から labelButton9 までのプロパティについては、出力ログレベルが 10～20 の場合、タスクログに出力される詳細は応答結果によって異なります。
- mailSubject または mailBody プロパティ内で機種依存文字を使用している場合や、文字セット間で互換性のない文字を使用している場合、その文字は疑問符（?）または他の文字で置き換えられます。この場合は、メールの文字を変更するか、エンコードを変更します。
- 次の文字は正しく変換されない可能性があります。

  ~, ¥, ＼, ～, ∥, －, ¢, £, ¬

- タスクを停止または強制停止した場合でも、［失敗した次のステップからリトライ］を実行すると応答入力なしで後続ステップが実行されます。後続ステップの実行を防止するため、［サービス作成］または［サービス編集］画面の［有効なアクション］で、リトライアクションを無効にすることを推奨します。
- 応答入力可能ユーザーを指定する場合の注意事項を次に示します。
  - responseUser プロパティに存在しないユーザーのみを指定してタスクを実行、またはタスク実行ユーザーのみを指定した場合、どのユーザーを用いても応答できなくなります。この場合、タスクを停止または強制停止してください。

- ［Service Builder Debug］画面においても、タスク実行ユーザーで応答入力することができません。［Service Builder Debug］画面でユーザー応答待ち部品の後続部品を実行するには、responseUser プロパティに値を指定しないか、ドライランモードでユーザー応答待ち部品を実行してください。

終了コード

ユーザー応答待ち部品では、次の終了コードが生成されます。

終了コード	説明
0-9	labelButton1 から labelButton9 までのプロパティに対応する終了コードを返します。応答待ちの間にタイムアウトが発生した場合、timeOutDefault プロパティに指定されている値が終了コードとして返されます。したがって、終了コードの意味は、部品を使用しているサービステンプレートによって異なります。
10-63	応答待ちの間にタイムアウトが発生した場合、timeOutDefault プロパティに指定されている値が終了コードとして返されます。
65	Ops Center Automator サーバとの接続に失敗しました。例えば、部品の実行時に Ops Center Automator サーバが停止した可能性があります。
66	次のユーザーが Ops Center Automator ユーザーにマッピングされています。 ・ Administrators group に属さないユーザー。 ・ UAC が有効な環境で、Administrators group に属するビルトイン Administrator 以外のユーザー。
68	対象のジョブ実行 ID についての情報がありません。
69	タスク処理エンジンの環境変数を取得できません。
80	タスクの実行が停止しました。
81	部品が不正な状態で呼び出されました。
82	タスク処理エンジンからの要求メッセージを正しく解析できません。
83	Ops Center Automator サーバの環境が破損しています。
84	指定された部品についての情報を取得できません。
86	［応答入力］ダイアログのマッピングパラメーターに設定した値は、プロパティの制約条件に反しています。
127	そのほかのエラーが発生しました。

プロパティリスト

ユーザー応答待ち部品では、次のプロパティを使用できます。

プロパティキー	プロパティ名	説明	デフォルト値	入出力タイプ	必須
toAddress	TO メールアドレス	受信者のメールアドレスを 1,024 文字以内で指定し、TO 属性に入力します。複数のアドレスを指定す	--	入力	任意

プロパティキー	プロパティ名	説明	デフォルト値	入出力タイプ	必須
		る場合、各アドレスをコンマで区切ります。			
ccAddress	CC メールアドレス	受信者のメールアドレスを 1,024 文字以内で指定し、CC 属性に入力します。複数のアドレスを指定する場合、各アドレスをコンマで区切ります。	--	入力	任意
bccAddress	BCC メールアドレス	受信者のメールアドレスを 1,024 文字以内で指定し、BCC 属性に入力します。複数のアドレスを指定する場合、各アドレスをコンマで区切ります。	--	入力	任意
mailSubject	メール件名	メールの件名行を 256 文字以内で指定します。	--	入力	任意
mailBody	メール本文	メールの本文テキストを 1,024 文字以内で指定します。	--	入力	任意
encodeType	エンコード種別	次のうちからメールのエンコードを 1 つ指定します。 • us-ascii • iso-2022-jp • shift_jis • euc-jp • utf-8	utf-8	入力	任意
dialogText	応答入力画面	［応答入力］ダイアログに表示される情報を指定します。この情報はテキストまたは HTML 形式で指定できます。	--	入力	必須
responseTimeOut	応答タイムアウト時間	応答待ちでタイムアウトが発生するまでの時間を、1～20,160（分単位）で指定します。	1440	入力	必須
timeOutDefault	タイムアウトデフォルト値	応答待ちでタイムアウトが発生したときに返される終了コードを指定します。タイムアウト時間が経過すると、終了コードとしてこの値が返されます。例えば、0 が指定されていてタイムアウト期間が経過すると、labelButton0 プロパティに関連付けられているボタンに対応する処理が実行されます。0～63 の範囲の終了コードを指定します。	0	入力	必須
labelButton0	ボタン 0 表示ラベル	終了コード 0 を生成する応答のボタンのラベルを、最大 256 文字で指定します。ユーザーの運用上の要求に応じたボタンを、［応答入力］ダイアログに表示できます。	--	入力	任意
labelButton1	ボタン 1 表示ラベル	終了コード 1 を生成する応答のボタンのラベルを、最大 256 文字で	--	入力	任意

プロパティキー	プロパティ名	説明	デフォルト値	入出力タイプ	必須
		指定します。ユーザーの運用上の要求に応じたボタンを、[応答入力] ダイアログに表示できます。このプロパティを省略すると、対応するボタンは表示されません。			
labelButton2	ボタン 2 表示ラベル	終了コード 2 を生成する応答のボタンのラベルを、最大 256 文字で指定します。ユーザーの運用上の要求に応じたボタンを、[応答入力] ダイアログに表示できます。このプロパティを省略すると、対応するボタンは表示されません。	--	入力	任意
labelButton3	ボタン 3 表示ラベル	終了コード 3 を生成する応答のボタンのラベルを、最大 256 文字で指定します。ユーザーの運用上の要求に応じたボタンを、[応答入力] ダイアログに表示できます。このプロパティを省略すると、対応するボタンは表示されません。	--	入力	任意
labelButton4	ボタン 4 表示ラベル	終了コード 4 を生成する応答のボタンのラベルを、最大 256 文字で指定します。ユーザーの運用上の要求に応じたボタンを、[応答入力] ダイアログに表示できます。このプロパティを省略すると、対応するボタンは表示されません。	--	入力	任意
labelButton5	ボタン 5 表示ラベル	終了コード 5 を生成する応答のボタンのラベルを、最大 256 文字で指定します。ユーザーの運用上の要求に応じたボタンを、[応答入力] ダイアログに表示できます。このプロパティを省略すると、対応するボタンは表示されません。	--	入力	任意
labelButton6	ボタン 6 表示ラベル	終了コード 6 を生成する応答のボタンのラベルを、最大 256 文字で指定します。ユーザーの運用上の要求に応じたボタンを、[応答入力] ダイアログに表示できます。このプロパティを省略すると、対応するボタンは表示されません。	--	入力	任意
labelButton7	ボタン 7 表示ラベル	終了コード 7 を生成する応答のボタンのラベルを、最大 256 文字で指定します。ユーザーの運用上の要求に応じたボタンを、[応答入力] ダイアログに表示できます。このプロパティを省略すると、対応するボタンは表示されません。	--	入力	任意
labelButton8	ボタン 8 表示ラベル	終了コード 8 を生成する応答のボタンのラベルを、最大 256 文字で指定します。ユーザーの運用上の要求に応じたボタンを、[応答入	--	入力	任意

プロパティキー	プロパティ名	説明	デフォルト値	入出力タイプ	必須
		力］ダイアログに表示できます。このプロパティを省略すると、対応するボタンは表示されません。			
labelButton9	ボタン 9 表示ラベル	終了コード 9 を生成する応答のボタンのラベルを、最大 256 文字で指定します。ユーザーの運用上の要求に応じたボタンを、［応答入力］ダイアログに表示できます。このプロパティを省略すると、対応するボタンは表示されません。	--	入力	任意
responseUser	応答入力可能ユーザー	応答入力可能なユーザー※1 を、最大 1,024 文字で指定します。使用できる文字は、半角英数字および記号（!#$%&'()*+-.=@¥^_|）です。値は大文字小文字を区別しません。複数のユーザーを指定する場合、半角コンマで区切ります。タスク実行ユーザーは、指定されていても応答入力できません。空（デフォルト）の場合、タスク実行ユーザーを含むすべてのユーザーが応答入力できます。※2	--	入力	任意

注※1　Common Services で認証された次のユーザーを指定します。

認証種別	指定方法
ユーザーディレクトリ（Active Directory）	ログイン時に指定するユーザー ID
ユーザーディレクトリ（LDAP サーバ）	
ローカルユーザー	
ID プロバイダー（OIDC）	UPN 形式のユーザー ID
ID プロバイダー（SAML）	AD FS の要求発行ポリシーの設定、および Common Services の NameID フォーマットに指定されているフォーマットのユーザー ID

詳細は、『Hitachi Ops Center インストールガイド』を参照してください。

注※2　responseUser プロパティの最小長を 1 以上に設定すると、空（デフォルト値）を設定できなくなります。本プロパティを使用する場合は、応答入力可能ユーザーの設定を必須とするため、プロパティの最小長を 1 以上に設定することを推奨します。詳細は、「4.5 プロパティ設定を指定する」を参照してください。

dialogText プロパティに指定できる HTML タグと属性

dialogText プロパティの表示内容を HTML 形式で指定するとき、次の表に記載されているタグを使用できます。

タグ	属性	ノートと制約条件
アンカータグ(<a>)		<a>タグを使用する際は、対象の属性は空白にしておく必要がありま

タグ	属性	ノートと制約条件
		す。このタグを指定しない場合、ページは［応答入力］ダイアログに読み込まれます。
	href	-
	target	「_blank.」と指定します。
ボールドタグ(<b>)		-
ブレークタグ( )		-
フォントタグ(<font>)	color	-
	face	-
	size	-
イタリックタグ(<i>)		-
下線タグ(<u>)		-
フォームタグ(<form>)		-
インプットタグ(<input>)		フォームタグで囲むことが一般的ですが、送る必要がないため、フォームタグで囲む必要はありません。
	name	サービスプロパティキーを指定する場合、応答を待っている間に、［応答入力］ダイアログで指定したサービスプロパティのマッピングパラメーターを変更できます。
	type	"text"、"checkbox"、"radio"を指定できます。
	value	type 属性が"checkbox"または"radio"の場合、value 属性を設定します。チェックボックスまたはラジオボタンで選択した項目の value 属性に設定した値は、name 属性で指定されたサービスプロパティのマッピングパラメーターになります。
セレクトタグ(<select>)	name	サービスプロパティキーを指定する場合、応答を待っている間に、［応答入力］ダイアログで指定したサービスプロパティのマッピングパラメーターを変更できます。
オプションタグ(<option>)		セレクトタグを使用して囲みます。
	value	セレクトメニューあるいはリストボックスで選択した項目の value 属性で設定した値は、セレクトタグの name 属性で指定されたサービスプロパティのマッピングパラメーターになります。

B.6 ターミナル接続部品

端末の接続を確立するために使用されます。

ターミナル接続部品により、Telnet または SSH を使用し、認証を実行して、操作対象機器へ接続できます。

Telnet で接続するときは、必要に応じてユーザー ID とパスワードを設定します。SSH 接続の場合は、パスワード認証、公開キー認証、またはキーボードインタラクティブ認証を認証方式として選択できます。部品のプロパティ、またはエージェントレス接続先定義ビューで、次の情報を設定する必要があります。

- 認証方式（パスワード認証、公開キー認証、またはキーボードインタラクティブ認証）
- パスワード認証に必要な情報（ユーザー ID とパスワード）
- 公開キー認証に必要な情報（ユーザー ID）
- キーボードインタラクティブ認証に必要な情報（ユーザー ID とパスワード）

ターミナルコマンド実行部品で指定されたコマンドは、ターミナル接続部品により認証されているユーザーの権限で実行されます。root 権限でコマンドを実行するには、ユーザーを root 権限に昇格するコマンドをターミナルコマンド実行部品で実行する必要があります。

実行の前提条件

- 部品は、protocol プロパティに指定されているプロトコルを使用して、Ops Center Automator サーバと通信します。
- Telnet で接続する場合、操作対象機器がユーザー ID とパスワードの入力を求めるとき、部品によって検出されます。必要に応じて以下のファイルの 1 つを設定します。両方のファイルを設定すると、接続先プロパティファイル（*connection-destination-name*.properties）に設定されている値が Ops Center Automator により使用されます。
  - 接続先プロパティファイル（*connection-destination-name*.properties）の telnet.prompt.account と telnet.prompt.password
  - プロパティファイル（config_user.properties）の plugin.terminal.prompt.account と plugin.terminal.prompt.password

注意事項

- 部品は、readWaitTime プロパティで指定されている時間だけ、標準出力を待ちます。標準出力からの出力が停止した後で、readWaitTime で指定されている時間が経過すると、部品は異常終了します。部品を使用する前に、readWaitTime プロパティの値が適切なことを確認してください。
- 標準出力へ出力された値が、promptPattern プロパティに指定されている正規表現パターンと一致する場合、部品はただちに終了します。
- Telnet を使用して操作対象機器への通信を確立した後で、部品は標準出力および標準エラー出力を、プロパティファイル（config_user.properties）の telnet.connect.wait プロパティに設定されている時間だけ待ちます。接続先サービスが Web サーバ、または標準出力か標準エラー出力を生成しない他のエンティティである場合、接続先プロパティファイル（*connection-destination-name*.properties）の telnet.noStdout.port.list property で

設定されているサービスのポート番号を設定します。ポート番号を設定すると、部品は標準出力または標準エラー出力を待たずに実行を終了します。

- 部品の実行中にタスクの実行が停止した場合、タスクの状態はターミナル接続部品の処理が終了したときに「失敗」または「正常終了」になります。その後で、セッションとトークンは破棄されます。部品の実行が終了した後のステップとタスクの状態は、ステップの終了コードおよび後続ステップの実行条件によって異なります。後続ステップ実行条件は、［ステップ作成］ダイアログまたは［ステップ編集］ダイアログで設定できます。
- ターミナル接続部品は、Telnet 認証が失敗した場合でも接続を維持します。接続を終了するには、ターミナル切断部品を実行する必要があります。ただし、タスクが「失敗」または「正常終了」状態に移行した場合、接続は自動的に終了され、ターミナル切断部品を実行する必要はありません。
- ターミナル接続部品の標準出力と標準エラー出力は、Ops Center Automator ステップの標準出力として出力されます。標準出力および標準エラー出力のサイズは、Ops Center Automator が受信した合計バイト数です。Telnet サーバまたは SSH サーバが、改行文字 LF を CR+LF に置き換えるよう構成されている場合、それぞれの改行文字を 2 バイトとして計算してください。処理結果の標準出力と標準エラー出力の合計が 100KB を超える場合、製品サポートの対象外となります。標準出力と標準エラー出力の合計が 100KB を超えないことを確認してください。
- ターミナル接続部品は、Telnet 接続での認証エラーを検出できません。このため、stdoutPattern1 から stdoutPattern3 までのいずれかに、標準出力および標準エラー出力の認証エラーを検出する正規表現パターンを指定してください。
- ターミナル接続部品のバージョンが 02.00.00 より前の場合、「outputCondition」に「patternMatch」が設定されます。ただし、バージョン 02.00.00 の場合、「outputCondition」のデフォルト値は「always」です。ターミナル接続部品をバージョンアップする場合、この点に注意してください。

終了コード

ターミナル接続部品では、次の終了コードが生成されます。

終了コード	説明
0～63	標準出力または標準エラー出力が、returnCodePattern プロパティで指定されている正規表現パターンと一致した場合、部品は returnCode プロパティに指定されている終了コードを返します。標準出力および標準エラー出力が、returnCodePattern プロパティで指定されているパターンと一致しない場合、部品は defaultReturnCode プロパティに指定されている終了コードを返します。したがって、終了コードの意味は、部品を使用しているサービステンプレートによって異なります。
65	Ops Center Automator サーバとの接続に失敗しました。例えば、部品の実行時に Ops Center Automator サーバが停止した可能性があります。
66	次のユーザーが Ops Center Automator ユーザーにマッピングされています。 ・ Administrators group に属さないユーザー。 ・ UAC が有効な環境で、Administrators group に属するビルトイン Administrator 以外のユーザー。
68	対象のジョブ実行 ID についての情報がありません。

終了コード	説明
69	タスク処理エンジンの環境変数を取得できません。
70	操作対象機器との接続に失敗しました。
76	接続のタイムアウトが発生しました。
77	操作対象機器のホスト名を解決できません。
78	SSH で接続するとき、操作対象機器での認証が失敗しました。
80	タスクの実行が停止しました。
81	部品が不正な状態で呼び出されました。
82	タスク処理エンジンからの要求メッセージを正しく解析できません。
83	Ops Center Automator サーバの環境が破損しています。
84	指定された部品についての情報を取得できません。
86	指定されたプロパティの値が無効です。
87	標準出力と標準エラー出力がタイムアウトしました。
88	トークンの最大数（タスクごとに 99）に達しました。標準出力と標準エラー出力の合計が 100KB を超えました。
127	そのほかのエラーが発生しました。

プロパティリスト

ターミナル接続部品では、次のプロパティを使用できます。

プロパティキー	プロパティ名	説明	デフォルト値	入出力タイプ	必須
destinationHost	対象機器	操作対象機器の IPv4 アドレス、IPv6 アドレス、またはホスト名を、最大 1,024 文字で指定します。複数の IP アドレスやホスト名を指定することはできません。	--	入力	必須
protocol	プロトコル	操作対象機器への接続に使用するプロトコルを指定します。次のプロトコルを指定できます。 ・ Telnet ・ SSH	Telnet	入力	任意
credentialType	認証種別	コマンドまたはスクリプトの実行時に使用する認証タイプとして、次のどちらかを指定します。 **Destination** ［エージェントレス接続先定義］ビューに設定されている認証情報を使用するには、このオプションを指定します。［destination］を指定すると、Ops Center Automator ログインユーザーの IP アドレスに従って、接続先定義で設定されている Telnet また	--	入力	必須

プロパティキー	プロパティ名	説明	デフォルト値	入出力タイプ	必須
		はSSHの認証情報が適用されます。認証情報に関連するプロパティは省略できます（account、password、suPassword、publicKeyAuthentication、およびkeyboardInteractiveAuthentication）。 **Property** 以下のプロパティに指定された値を認証情報として使用するには、このオプションを指定します。 ・ account ・ password ・ suPassword ・ publicKeyAuthentication ・ keyboardInteractiveAuthentication			
account	ユーザー ID	操作対象機器へのログインに使用するユーザー ID を最大 256 文字で指定します。 次の両方の条件が当てはまる場合、このプロパティは必須です。 ・ protocol プロパティに SSH が指定されている。 ・ credentialType プロパティに property が指定されている。	--	入力	任意
password	パスワード	操作対象機器へのログインに使用するパスワードを最大 256 文字で指定します。次の条件がすべて当てはまる場合、このプロパティは必須です。 ・ protocol プロパティに SSH が指定されている。 ・ credentialType プロパティに property が指定されている。 ・ publicKeyAuthentication プロパティに false が指定されている。 操作対象機器の OS が Linux で、publicKeyAuthentication プロパティに true が指定されている場合、ここに指定した値は無視されます。ただし、参照用に reserved.terminal.password 予	--	入力	任意

プロパティキー	プロパティ名	説明	デフォルト値	入出力タイプ	必須
		約プロパティへ値を設定できます。			
suPassword	管理者のパスワード	ユーザーを root 権限へ昇格するため必要なパスワードを、最大 256 文字で指定します。コマンド行またはターミナルコマンド実行部品に reserved.terminal.suPassword プロパティを指定すると、suPassword プロパティの値はそのプロパティに割り当てられます。	--	入力	任意
publicKeyAuthentication	SSH 公開鍵認証設定	操作対象機器の OS が Linux の場合、公開鍵認証を使用するかどうかに応じて、以下のどちらかを指定します。値は大文字小文字を区別しません。値を指定しない場合は false とみなされます。操作対象機器で Windows が実行されている場合、このプロパティを省略できます。 **true** 公開鍵認証を使用するには、このオプションを指定します。 **false** パスワード認証またはキーボードインタラクティブ認証を使用するには、このオプションを指定します。	false	入力	任意
keyboardInteractiveAuthentication	キーボードインタラクティブ認証設定	Linux 環境で SSH キーボードインタラクティブ認証を使用するかどうかを制御します。接続先のオペレーティングシステムが Linux の場合、システムはキーボードインタラクティブ認証を使用するかどうかを切り替えます。このプロパティを true に設定した場合、キーボードインタラクティブ認証を使用します。このプロパティを false に設定した場合、キーボードインタラクティブ認証を使用しません。このプロパティは大文字小文字を区別しません。このプロパティは、publicKeyAuthentication を false に設定している場合のみ有効です。このプロパティが存在しない（以前の部品バージョンの場合）または値が指定されていない場合、このプロパティは false とみなされます。	false	入力	任意

プロパティキー	プロパティ名	説明	デフォルト値	入出力タイプ	必須
Port	ポート番号	操作対象機器への接続に使用するポート番号を指定します。	--	入力	任意
charset	文字セット	操作対象機器の標準入力への書き込みと、標準出力および標準エラー出力からの読み出しに使用される文字セットを指定します。操作対象にログインするユーザーのものと同じ文字セットを指定します。文字セットの名前では、大文字と小文字は区別されません。次の文字セットを指定できます。 ・ EUC-JP ・ eucjp ・ ibm-943C ・ ISO-8859-1 ・ MS932 ・ PCK ・ Shift_JIS ・ UTF-8 ・ windows-31j	--	入力	任意
lineEnd	行端文字	ターミナル接続部品の protocol プロパティに Telnet が指定されている場合、account および password プロパティに指定されている値に追加される改行文字を指定します。次の文字を指定できます。 ・ CR ・ LF ・ CRLF 改行文字として 0x0D を使用するには、CR を指定します。0x0A を使用するには LF を、0x0D0A を使用するには CRLF を指定します。	CR	入力	任意
promptPattern	プロンプトパターン	標準出力と標準エラー出力でプロンプトの検出に使用する正規表現パターンを、最大 1,024 文字で指定します。このプロパティは、操作対象機器が接続を確立した後で、コマンドを実行できる状態を検出するために使用されます。PCRE 準拠の形式でパターンを指定します。指定されている正規表現パターンと出力とが一致すると、部品はただちに終了します。出力がパターンと一致しない場合、標準出力または標準	--	入力	必須

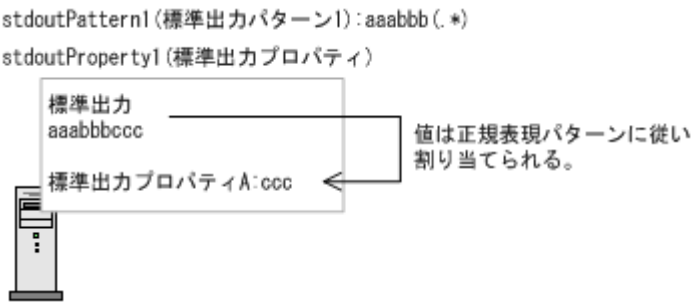
プロパティキー	プロパティ名	説明	デフォルト値	入出力タイプ	必須
		エラー出力に最後の出力が行われてから、readWaitTime プロパティに設定されている時間が経過すると、部品は異常終了します。			
readWaitTime	標準出力待ち時間	操作対象機器にログインするとき、標準出力または標準エラー出力に出力が行われてから、次の出力まで待つ時間を指定します。タイムアウト時間を、1～86,400,000（ミリ秒）の範囲で指定します。	60000	入力	任意
token	トークン文字列	セッションを識別するトークン文字列は、このプロパティに出力されます。このプロパティへ出力される文字列は、ターミナルコマンド実行部品およびターミナル切断部品の token プロパティに指定できます。	--	出力	任意
outputCondition	出力条件	標準出力プロパティ 1～3 に出力する条件を指定します。次の値を指定できます。 • always -- 指定されたパターンと一致しなくても、null 文字を出力します。 • patternMatch -- 標準出力パターン 1～3 に一致する場合だけ出力します。 出力プロパティに出力がない場合、マッピングされたサービスプロパティも更新されません。	always	入力	任意
stdoutPattern1	標準出力パターン 1	標準出力および標準エラー出力から stdoutProperty プロパティへ出力される正規表現パターンを、最大 1,024 文字で指定します。PCRE 準拠の形式でパターンを指定します。 1,024 文字を超えた場合、1,025 文字目とそれ以降の文字は切り捨てられます。	--	入力	任意
stdoutProperty1	標準出力プロパティ 1	stdoutPattern1 プロパティから抽出された文字列は、このプロパティに出力されます。	--	出力	任意
stdoutPattern2	標準出力パターン 2	標準出力および標準エラー出力から stdoutProperty プロパティへ出力される正規表現パターンを、最大 1,024 文字で指定します。PCRE 準拠の形式でパターンを指定します。		入力	任意

プロパティキー	プロパティ名	説明	デフォルト値	入出力タイプ	必須
		1,024 文字を超えた場合、1,025 文字目とそれ以降の文字は切り捨てられます。			
stdoutProperty2	標準出力プロパティ 2	stdoutPattern2 プロパティから抽出された文字列は、このプロパティに出力されます。	--	出力	任意
stdoutPattern3	標準出力パターン 3	標準出力および標準エラー出力から stdoutProperty プロパティへ出力される正規表現パターンを、最大 1,024 文字で指定します。PCRE 準拠の形式でパターンを指定します。 1,024 文字を超えた場合、1,025 文字目とそれ以降の文字は切り捨てられます。	--	入力	任意
stdoutProperty3	標準出力プロパティ 3	stdoutPattern3 プロパティから抽出された文字列は、このプロパティに出力されます。	--	出力	
defaultReturnCode	デフォルト戻り値	returnCodePattern プロパティに指定されている正規表現パターンと、標準出力および標準エラー出力とが一致しない場合、終了コードとして返す値を指定します。0～63 の範囲の値を指定します。	0	入力	任意
returnCodePattern	戻り値判定パターン	標準出力と標準エラー出力の正規表現パターンを、最大 1,024 文字で指定します。PCRE 準拠の形式でパターンを指定します。標準出力および標準エラー出力が、指定されているパターンと一致した場合、部品は returnCode プロパティに指定されている値を返します。	--	入力	任意
returnCode	戻り値	標準出力と標準エラー出力が、returnCodePattern プロパティに設定されているパターンと一致したとき、部品によって返される終了コードを指定します。0～63 の範囲の値を指定します。このプロパティを省略した場合、defaultReturnCode プロパティに指定されている値が部品から返されます。	--	入力	任意

stdoutPattern プロパティと stdoutProperty プロパティの使用例

stdoutPattern プロパティを使用して、標準出力に出力する値を抽出し、stdoutProperty プロパティに格納することができます。以下の図は、stdoutPattern1 に「aaabbb(.*)」を指定した場合のデータフローを示しています。

stdoutPattern および stdoutProperty プロパティの使用例を、以下に示します。



stdoutPattern1 で定義されているように、標準出力「aaabbbccc」では、「aaabbb」以降の値（この場合は「ccc」）が抽出されます。抽出された値は stdoutProperty1 プロパティに格納されます。

部品のプロパティが複数の場所に設定されている場合の優先順位

部品のプロパティに関連する情報は、接続先プロパティファイル（*connection-destination-name*.properties）やプロパティファイル（config_user.properties）にも設定できます。特定のプロパティの値が複数の場所で設定されている場合、次の優先順位が適用されます。

設定	場所	プロパティキー	優先順位	デフォルト値
Telnet ポート番号	部品のプロパティ	port	1	--
	接続先プロパティファイル（*connection-destination-name*.properties）	telnet.port	2	--
	プロパティファイル（config_user.properties）	telnet.port.number	3	23
SSH ポート番号	部品のプロパティ	port	1	--
	接続先プロパティファイル（*connection-destination-name*.properties）	ssh.port	2	--
	プロパティファイル（config_user.properties）	ssh.port.number	3	22
文字セット名	部品のプロパティ	charset	1	--
	接続先プロパティファイル（*connection-destination-name*.properties）	terminal.charset	2	--

部品のプロパティと接続先プロパティファイル（*connection-destination-name*.properties）のどちらにも値が設定されていない場合、UTF-8 が設定されます。

ターミナル接続部品の使用例

Telnet 認証エラーを判定する例

部品のプロパティを使用して次のような処理を実現する例について、以下に説明します。

- ログインが成功した場合に 0 を返す。
- ログインが失敗した場合に 1 を返す。
- ログインが成功した場合、最後のログインの日時と、接続元についての情報を、stdoutProperty1 プロパティに格納する。

この処理を実現するため、部品のプロパティに指定できる値の例を、次の表に示します。

プロパティキー	指定する値の例	指定する値の意味
promptPattern	^¥[prompt¥] | ^Login incorrect	標準出力の内容が[prompt]または Login incorrect と一致した場合、部品は終了し、終了コードを決定します。
stdoutPattern1	^Last login:(.*)	標準出力で、Last login:に続く文字列が、stdoutProperty1 プロパティに格納されます。
defaultReturnCode	0	returnCodePattern プロパティに指定されている値と、標準出力の内容とが一致しない場合、0 が返されます。
returnCodePattern	^ Login incorrect	標準出力の内容が Login incorrect と一致した場合、部品は returnCode プロパティに指定されている終了コードを返します。
returnCode	1	returnCodePattern プロパティに指定されている値と、標準出力の内容とが一致した場合、部品は 1 を返します。

先に列挙されたプロパティを持つ部品が次のような標準出力を受け取ったときの動作を、以下に示します。

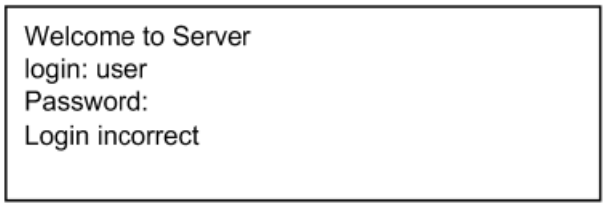
```
Welcome to Server
login:user
password:

Login OK
Last login: Mon Mar 18 13:21:13 2013 from ServerA
[prompt]>
```

これは、ログインが成功したときの例です。

標準出力の内容と、promptPattern プロパティに指定されている値とが一致するため、ターミナル接続部品は終了コードを決定します。この場合、標準出力と、returnCodePattern プロパティに指定されている値とが一致しないため、部品は defaultReturnCode プロパティに指定されている値（0）を終了コードとして返します。

stdoutPattern1 プロパティによって抽出された文字列（Mon Mar 18 13:21:13 2013 fromServerA）は、stdoutProperty1 プロパティに格納されます。



これは、ログインが失敗したときの例です。

標準出力の内容と、promptPattern プロパティに指定されている値とが一致するため、ターミナル接続部品の終了コードが決定されます。この場合、終了コードと、returnCodePattern プロパティに指定されている値とが一致するため、部品は returnCode プロパティに指定されている値（1）を返します。

SSH の使用時に認証エラーが発生したかどうかをチェックする

プロトコルとして SSH を使用する場合、ターミナル接続部品の終了コードを調べて、認証エラーが発生したかどうかをチェックできます。

認証エラーの検出には、エージェントレス接続先定義ビューの認証情報設定、またはターミナル接続部品の認証関連のプロパティ（account、password、および publicKeyAuthentication）を使用します。この処理では、エージェントレス接続先定義ビューに設定されているスーパーユーザーパスワードや、ターミナル接続部品の suPassword プロパティは使用しません。

認証エラーが検出された場合、部品はコード 78 を返します。credentialType プロパティに宛先が指定され、エージェントレス接続先定義ビューの認証情報が正しく設定されていない場合、部品の終了コードは 70 になることに注意してください。

HTTP サーバなど、標準出力を生成しないサービスへ接続する例

標準出力を生成しないサービスへ接続する例について、以下に説明します。この例では、接続先プロパティファイル（*connection-destination-name*.properties）の telnet.noStdout.port.list プロパティに 80 が指定されていることを想定しています。

この場合、以下のプロパティに指定されている値は無視され、部品は終了コード 0 を返します。

- credentialType
- account
- password
- suPassword
- publicKeyAuthentication
- keyboardInteractiveAuthentication
- charset

- lineEnd
- promptPattern
- readWaitTime
- stdoutPattern1 から stdoutPattern3 まで
- defaultReturnCode
- returnCodePattern
- returnCode

B.7 ターミナルコマンド実行部品

ターミナルコマンド実行部品は、ターミナル接続部品により接続されている接続先ホストでコマンドを実行します。

機能

この部品を使用すると、ターミナル接続部品により接続されている操作対象機器で指定のコマンドを実行できます。ターミナルコマンド実行部品で指定されるコマンドは、ターミナル接続部品で認証されているユーザー権限で実行されます。root 権限でコマンドを実行するには、ユーザーを root 権限に昇格するコマンドをターミナルコマンド実行部品で実行する必要があります。

実行の前提条件

- ターミナル接続部品の protocol プロパティに指定されているプロトコルが、Ops Center Automator サーバとの通信に使用されます。
- ターミナル接続部品により、操作対象機器との接続が確立されている必要があります。

注意事項

- 部品は、readWaitTime プロパティで指定されている時間だけ、標準出力を待ちます。標準出力からの出力が停止した後で、readWaitTime で指定されている時間が経過すると、部品は異常終了します。部品を使用する前に、readWaitTime プロパティの値が適切なことを確認してください。部品のタイムアウト後に出力された情報はすべて破棄されます。
- 標準出力へ出力された値が、promptPattern プロパティに指定されている正規表現パターンと一致する場合、部品はただちに終了します。
- コマンドからの情報が 1 ページずつ出力される場合、システムは標準出力が停止したと判断します。その後で readWaitTime プロパティに指定されている時間が経過すると、部品は異常終了します。ターミナルコマンド実行部品により実行されるコマンドが、結果を 1 ページずつ出力するよう構成されていないことを確認してください。
- echo されたコマンドラインも、標準出力へ出力されます。必要なら、コマンドが echo を返さないように構成してください。
- 部品の実行中にタスクの実行が停止した場合、ターミナルコマンド実行部品の処理が終了したときに、タスクの状態が「失敗」または「正常終了」になります。その後で、セッションとトークンは破棄されます。部品の実行が終了した後のステップとタスクの状態は、ステップの終了コードおよび後続ステップの実行条件によって異なります。後続ステップ実行条件は、[ステップ作成] ダイアログまたは [ステップ編集] ダイアログで設定できます。
- ターミナルコマンド実行部品の標準出力と標準エラー出力は、Ops Center Automator ステップの標準出力として出力されます。標準出力および標準エラー出力のサイズは、Ops Center Automator が受信した合計バイト数です。Telnet サーバまたは SSH サーバが、改行文字 LF を

CR+LF に置き換えるよう構成されている場合、それぞれの改行文字を 2 バイトとして計算してください。処理結果の標準出力と標準エラー出力の合計が 100KB を超える場合、製品サポートの対象外となります。標準出力と標準エラー出力の合計が 100KB を超えないことを確認してください。

- commandLine プロパティに ASCII 以外の文字を指定する方法については、「B.1 汎用コマンド実行部品」を参照してください。
- ターミナル接続部品のバージョンが 02.00.00 より前の場合、「outputCondition」に「patternMatch」が設定されます。ただし、バージョン 02.00.00 の場合、「outputCondition」のデフォルト値は「always」です。ターミナル接続部品をバージョンアップする場合、この点に注意してください。

終了コード

ターミナルコマンド実行部品では、次の終了コードが生成されます。

終了コード	説明
0～63	標準出力および標準エラー出力が、returnCodePattern プロパティで指定されている正規表現パターンと一致した場合、部品は returnCode プロパティに指定されている終了コードを返します。ただし、出力がこのパターンと一致しない場合、部品は defaultReturnCode プロパティに指定されている終了コードを返します。したがって、終了コードの意味は、部品を使用しているサービステンプレートによって異なります。
65	Ops Center Automator サーバとの接続に失敗しました。例えば、部品の実行時に Ops Center Automator サーバが停止した可能性があります。
66	次のユーザーが Ops Center Automator ユーザーにマッピングされています。 ・ Administrators group に属さないユーザー。 ・ UAC が有効な環境で、Administrators group に属するビルトイン Administrator 以外のユーザー。
68	対象のジョブ実行 ID についての情報がありません。
69	タスク処理エンジンの環境変数を取得できません。
70	操作対象機器との接続に失敗しました。
80	タスクの実行が停止しました。
81	部品が不正な状態で呼び出されました。
82	タスク処理エンジンからの要求メッセージを正しく解析できません。
83	Ops Center Automator サーバの環境が破損しています。
84	指定された部品についての情報を取得できません。
86	指定されたプロパティの値が無効です。
87	標準出力と標準エラー出力がタイムアウトしました。
88	標準出力と標準エラー出力の合計が 100KB を超えました。
127	そのほかのエラーが発生しました。

プロパティリスト

ターミナルコマンド実行部品では、次のプロパティを使用できます。

プロパティキー	プロパティ名	説明	入出力タイプ
token※	トークン	ターミナル接続部品の token プロパティの値を指定します。	入力
commandLine	コマンドライン	操作対象機器で実行するコマンドまたはスクリプトの絶対パスを最大 1,024 文字で指定します。Ops Center Automator サーバと操作対象機器のオペレーティングシステムで、コマンドラインに入力可能な文字で指定します。コマンドライン内の環境変数を表す特殊文字はエスケープされません。特殊文字を文字列として処理するには、Windows ではパーセント（%）、Linux では円記号（¥）を使用して文字をエスケープします。 ユーザーに root 権限を与えるため、コマンドラインでスーパーユーザーパスワードを入力する必要がある場合、予約プロパティ reserved.terminal.suPassword を指定します。reserved.terminal.account、reserved.terminal.password、および reserved.terminal.suPassword 予約プロパティは、ターミナル接続部品用に設定されているトークン関連の認証情報を参照します。これらのプロパティが実際にどの認証情報を参照するかは、ターミナル接続部品の credentialType プロパティの設定によって異なります。 ・ credentialType プロパティに宛先が指定されている場合、これらの予約プロパティは接続先に定義されている認証情報を参照します。 ・ credentialType プロパティにプロパティが指定されている場合、これらの予約プロパティはターミナル接続部品の credentialType プロパティに指定されている認証情報を参照します。	入力
charset	文字セット	操作対象機器の標準入力への書き込みと、標準出力および標準エラー出力からの読み出しに使用される文字セットを指定します。操作対象にログインするユーザーのものと同じ文字セットを指定します。文字セットの名前では、大文字と小文字は区別されません。次の文字セットを指定できます。 ・ EUC-JP ・ eucjp ・ ibm-943C ・ ISO-8859-1 ・ MS932 ・ PCK ・ Shift_JIS ・ UTF-8 ・ windows-31j	入力

プロパティキー	プロパティ名	説明	入出力タイプ
lineEnd	行端文字	ターミナル接続部品の protocol プロパティに Telnet が指定されている場合、account および password プロパティに指定されている値に追加される改行文字を指定します。 次の文字を指定できます。 ・ CR ・ LF ・ CRLF 改行文字として 0x0D を使用するには、CR を指定します。0x0A を使用するには LF を、0x0D0A を使用するには CRLF を指定します。 デフォルト値は CR です。	入力
promptPattern※	プロンプトパターン	標準出力と標準エラー出力でプロンプトの検出に使用する正規表現パターンを、最大 1,024 文字で指定します。このプロパティは、操作対象機器が接続を確立した後で、コマンドを実行できる状態を検出するために使用されます。PCRE 準拠の形式でパターンを指定します。指定されている正規表現パターンと出力とが一致すると、部品はただちに終了します。出力がパターンと一致しない場合、標準出力または標準エラー出力に最後の出力が行われてから、readWaitTime プロパティに設定されている時間が経過すると、部品は異常終了します。	入力
readWaitTime	標準出力待ち時間	操作対象機器にログインするとき、標準出力または標準エラー出力に出力が行われてから、次の出力まで待つ時間を指定します。タイムアウト時間を、1～86,400,000（ミリ秒）の範囲で指定します。 デフォルト値は 60000 ミリ秒です。	入力
outputCondition	出力条件	標準出力プロパティ 1～3 に出力する条件を指定します。次の値を指定できます。 ・ always -- 指定されたパターンと一致しなくても、null 文字を出力します。 ・ patternMatch -- 標準出力パターン 1～3 に一致する場合だけ出力します。 出力プロパティに出力がない場合、マッピングされたサービスプロパティも更新されません。 デフォルト値は always です。	入力
stdoutProperty1	標準出力プロパティ 1	stdoutPattern1 プロパティから抽出された文字列は、このプロパティに出力されます。	出力
stdoutPattern1	標準出力パターン 1	stdoutProperty1 プロパティに出力する標準出力の正規表現パターンを最大 1,024 文字で指定します。正規表現パターンは PCRE に準拠する形式で指定します。 stdoutProperty1 プロパティにサービスプロパティのキーを指定し、stdoutPattern1 プロ	入力

プロパティキー	プロパティ名	説明	入出力タイプ
		パティを指定しない場合、commandLine プロパティに指定したコマンドまたはスクリプトの標準出力と標準エラー出力全体がサービスプロパティに割り当てられます。	
stdoutProperty2	標準出力プロパティ 2	stdoutPattern2 プロパティから抽出された文字列は、このプロパティに出力されます。	出力
stdoutPattern2	標準出力パターン 2	標準出力から stdoutProperty2 プロパティへ出力される正規表現パターンを、最大 1,024 文字で指定します。正規表現パターンは PCRE に準拠する形式で指定します。 stdoutProperty2 プロパティにサービスプロパティのキーを指定し、stdoutPattern2 プロパティを指定しない場合、commandLine プロパティに指定したコマンドまたはスクリプトの標準出力と標準エラー出力全体がサービスプロパティに割り当てられます。	入力
stdoutProperty3	標準出力プロパティ 3	stdoutPattern3 プロパティから抽出された文字列は、このプロパティに出力されます。	出力
stdoutPattern3	標準出力パターン 3	stdoutProperty3 プロパティに出力する標準出力の正規表現パターンを最大 1,024 文字で指定します。正規表現パターンは PCRE に準拠する形式で指定します。 stdoutProperty3 プロパティにサービスプロパティのキーを指定し、stdoutPattern3 プロパティを指定しない場合、commandLine プロパティに指定したコマンドまたはスクリプトの標準出力と標準エラー出力全体がサービスプロパティに割り当てられます。	入力
defaultReturnCode	デフォルト戻り値	returnCodePattern プロパティに指定されている正規表現パターンと、標準出力および標準エラー出力とが一致しない場合、終了コードとして返す値を指定します。0～63 の範囲の値を指定します。 デフォルト値は 0 です。	入力
returnCodePattern	戻り値判定パターン	標準出力と標準エラー出力の正規表現パターンを、最大 1,024 文字で指定します。 PCRE 準拠の形式でパターンを指定します。 標準出力および標準エラー出力が、指定されているパターンと一致した場合、部品は returnCode プロパティに指定されている値を返します。	入力
returnCode	戻り値	標準出力と標準エラー出力が、returnCodePattern プロパティに設定されているパターンと一致したとき、部品によって返される終了コードを指定します。0～63 の範囲の値を指定します。このプロパティを省略した場合、defaultReturnCode プロパティに指定されている値が部品から返されます。	入力

注※　必須プロパティです。

ターミナルコマンド実行部品の使用例

標準出力へエラーが出力されたとき、ターミナルコマンド実行部品を異常終了する例

次の表は、標準出力からエラー関連の情報を取得したとき、ターミナルコマンド実行部品を異常終了する例です。部品のプロパティを次のように設定します。

プロパティキー	指定する値の例	指定する値の意味
commandLine	configServer arg0 arg1 arg2	指定されたコマンドまたはスクリプトを実行します。
promptPattern	^¥[prompt¥]	標準出力の内容が[prompt]と一致した場合、部品は終了し、終了値を決定します。
stdoutPattern1	^Message:(.*)	標準出力で、Message:に続く文字列が、stdoutProperty1 プロパティに格納されます。
stdoutPattern2	^Error:(.*)	標準出力で、Error:に続く文字列が、stdoutProperty2 プロパティに格納されます。
stdoutPattern3	^ReturnCode:(.*)	標準出力で、Returncode:に続く文字列が、stdoutProperty3 プロパティに格納されます。
defaultReturnCode	0	標準出力の内容と、returnCodePattern プロパティに指定されている値とが一致しない場合、終了コード 0 が返されます。
returnCodePattern	^Error:	標準出力の内容が Error:と一致した場合、部品は returnCode プロパティに指定されている終了コードを返します。
returnCode	1	returnCodePattern プロパティに指定されている値と、標準出力の内容とが一致した場合、部品は終了コード 1 を返します。

先に列挙されたプロパティを持つ部品が次のような標準出力を受け取ったときの動作を、以下に示します。

```
configServer arg0 arg1 arg2
Message:command failed
Error:Permission Denied
ReturnCode:128
[prompt]>
```

標準出力の内容と、promptPattern プロパティに指定されている値とが一致するため、ターミナルコマンド実行部品はどの終了コードを返すかを判定します。標準出力と、returnCodePattern プロパティに指定されている値とが一致するため、部品は returnCode プロパティに指定されている値（1）を終了コードとして返します。

プロパティ stdoutPattern1 から stdoutPattern3 までから抽出された文字列は、プロパティ stdoutPrpoerty1 から stdoutProperty3 までに、次のように格納されます。

- stdoutProperty1: command failed
- stdoutProperty2: Permission Denied
- stdoutProperty3: 128

HTTP サーバ GET 要求を送信する例

HTTP サーバへ次のような要求を発行し、応答を検証する部品を構成する方法を、以下に示します。

```
GET /index.html HTTP/1.1
Host: ServerA
User-Agent: Ops Center Automator
Accept-Charset: UTF-8
```

HTTP サーバへ GET 要求を発行するには、要求メソッドの各行と要求ヘッダーを、ターミナルコマンド実行部品の commandLine プロパティに指定します。

要求の最後の行は空白にする必要があるため、ターミナルコマンド実行部品を 5 回実行する必要があります。部品の各インスタンスのプロパティに設定する値の例を、次の表に示します。

実行順序	commandLine に指定する値	lineEnd に指定する値	promptPattern に指定する値
1 回目	GET /index.html HTTP/1.1	CRLF	.*
2 回目	Host: ServerA	CRLF	.*
3 回目	User-Agent: Ops Center Automator	CRLF	.*
4 回目	Accept-Charset: UTF-8	CRLF	.*
5 回目	--（空行を追加します。値は指定しません）	CRLF	.*

HTTP サーバ要求は区切り文字として[CR]+[LF]を使用するため、lineEnd の値には CRLF を指定します。

1 回目から 4 回目までのターミナルコマンド実行部品の promptPattern プロパティには、空白文字とも一致する正規表現パターンを指定できます。

ターミナルコマンド実行部品の実行後も標準出力は継続するため、ターミナルコマンド実行部品による標準出力の末尾を検出できる正規表現を、promptPattern プロパティに指定します。

先に列挙されたプロパティを持つ部品が次のような標準出力を受け取ったときの動作を、以下に示します。

```
HTTP/1.1 200 OK
Date: Mon, 18 Mar 2013 10:19:20 GMT
Server: Cosminexus HTTP Server
Last-Modified: Sun, 31 Jul 2005 05:27:52 GMT
ETag: "2d000000012d48-f-3fd2b60590600"
Accept-Ranges: bytes
Content-Length: 15
Content-Type: text/html

<HTML></HTML>
```

標準出力の内容と、promptPattern プロパティに指定されている値とが一致するため、ターミナルコマンド実行部品は終了コードを決定します。

標準出力と、returnCodePattern プロパティに指定されている値とが一致する場合、部品の終了コードとして、returnCode プロパティに指定されている終了コードが返されます。

標準出力と、returnCodePattern プロパティに指定されている値とが一致しない場合、部品は defaultReturnCode プロパティに指定されている終了コードを返します。

B.8 ターミナル切断部品

ターミナル切断部品は、ターミナル接続部品により確立された、操作対象機器との接続を終了します。

実行の前提条件

- ターミナル切断部品は、ターミナル接続部品の protocol プロパティに指定されているプロトコルを、Ops Center Automator サーバとの通信に使用します。

注意事項

- 部品の実行中にタスクの実行が停止した場合、ターミナル切断部品の処理が終了したときに、タスクの状態が「失敗」または「正常終了」になります。部品の実行が終了した後のステップとタスクの状態は、ステップの終了コードおよび後続ステップの実行条件によって異なります。後続ステップ実行条件は、［ステップ作成］ダイアログまたは［ステップ編集］ダイアログで設定できます。
- 部品の実行中にタスクを強制終了した場合、標準出力からの読み出しとプロンプトの検出は取り消され、タスクの状態は「失敗」になります。その後で、セッションとトークンは破棄されます。この場合、［タスク詳細］ダイアログでステップの終了コードは 80 になります。タスクログへ出力される終了コードは、タスクが強制終了されたタイミングにより異なります。

終了コード

ターミナル切断部品では、次の終了コードが生成されます。

終了コード	説明
0	部品は正常に終了しました。接続がすでに閉じている場合でも部品は正常に終了します。
65	Ops Center Automator サーバとの接続に失敗しました。例えば、部品の実行時に Ops Center Automator サーバが停止した可能性があります。
66	次のユーザーが Ops Center Automator ユーザーにマッピングされています。 • Administrators group に属さないユーザー。 • UAC が有効な環境で、Administrators group に属するビルトイン Administrator 以外のユーザー。
68	対象のジョブ実行 ID についての情報がありません。
69	タスク処理エンジンの環境変数を取得できません。
80	タスクの実行が停止しました。
81	部品が不正な状態で呼び出されました。

終了コード	説明
82	タスク処理エンジンからの要求メッセージを正しく解析できません。
83	Ops Center Automator サーバの環境が破損しています。
84	指定された部品についての情報を取得できません。
86	指定されたプロパティの値が無効です。
127	そのほかのエラーが発生しました。

プロパティリスト

ターミナル切断部品では、次のプロパティを使用できます。

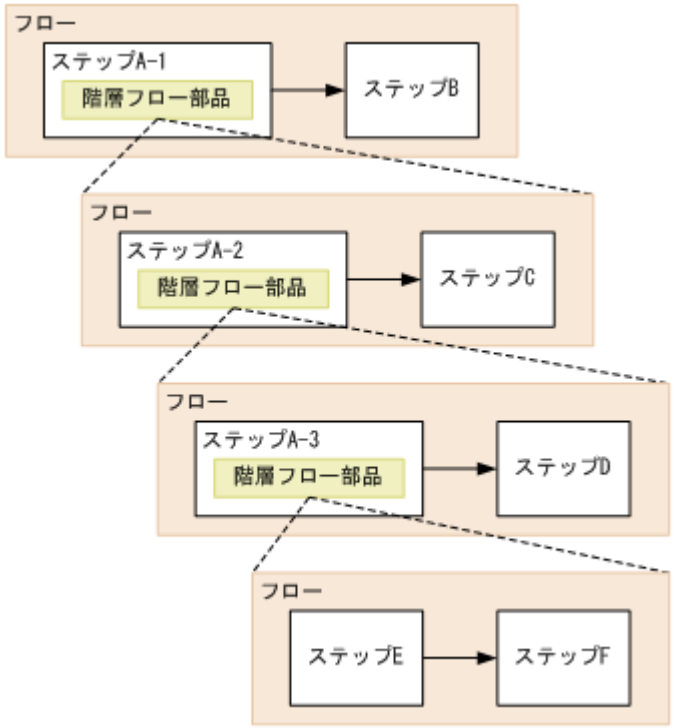
プロパティキー	プロパティ名	説明	入出力タイプ	必須
token	トークン	ターミナル接続部品の token プロパティの値を指定します。	入力	必須

B.9 階層フロー部品

階層フロー部品を使用して、フローの階層を定義することができます。

階層フロー部品を使用すると、フローを他のフロー内に定義することで階層フローを作成できます。最上位のフローのレベルを 1 として、最大 25 の階層レベルを定義できます。

次の図に、フローがどのように作成されるかを示します。



注意事項

- 部品の実行中にタスクの実行が停止した場合、タスクの状態は階層フロー部品内で実行されているステップが終了したときに「失敗」または「正常終了」になります。
- 階層フロー部品の終了コードは常に 0 です。階層フロー内のステップが異常終了しても、部品は終了コード 0 を返します。階層フロー部品の終了コードは、階層フローの構成ステップの終了コードを反映しません。

終了コード

階層フロー部品では、次の終了コードが生成されます。

終了コード	説明
0	部品は正常に終了しました。
1	下位の実行フローのステップが警告で終了しました。
2	下位の実行フローのステップが異常終了しました。

メモ　下位のフローのステップ実行が失敗した場合、タスクはその時点で終了しません。タスクの継続または停止の決定は、階層フロー部品ステップの後続ステップ実行条件を設定することによって行われます。

プロパティリスト

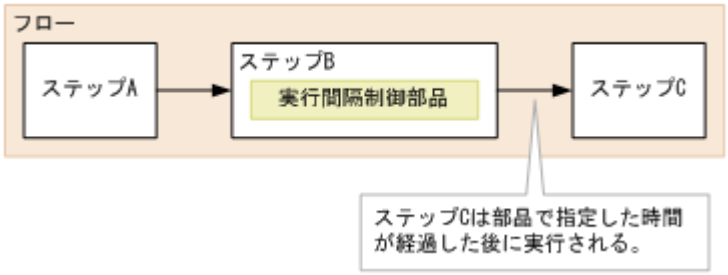
階層フロー部品では、次のプロパティを使用できます。

プロパティキー	プロパティ名	説明	入出力タイプ
errorStep	失敗したステップ	1 つ下位のフローで失敗したステップの ID が、半角コンマで区切って出力されます。	出力
returnValueOfErrorStep	失敗したステップの戻り値	1 つ下位のフローで失敗したステップの戻り値が、半角コンマ区切りの値で出力されます。	出力

B.10 実行間隔制御部品

実行間隔制御部品は、ステップ間の実行間隔を制御します。オペレータがプロセスの待機時間を実行間隔として指定します。Ops Center Automator は実行間隔が経過するまで待機してから、次のステップを実行します。実行間隔制御部品を使用すると、一定の間隔でステップを実行できます。

次の図に実行間隔制御部品の使用方法を示します。



注意事項

- 通信状態やその他の要因によって、実際の待機時間と部品に指定した待機時間との間に差が生じる可能性があります。
- サービスの実行時にプロパティ値を変更することはできません。フローの作成時に値を設定します。
- 入力プロパティにはリテラル文字のみを指定できます。サービスプロパティや予約済みプロパティの値をマッピングすることはできません。
- 部品の実行中にタスクの実行が停止した場合、タスクの状態は実行間隔制御部品の処理が終了したときに「失敗」または「正常終了」になります。

終了コード

実行間隔制御部品では、次の終了コードが生成されます。

終了コード	説明
0	部品は正常に終了しました。接続がすでに閉じている場合でも部品は正常に終了します。
18	部品の実行中にタスクが強制的に終了しました。
1～17、19 以上	部品が異常終了しました。**hcmds64getlogs** コマンドを使用してログ情報を取得し、問題を特定してください。

プロパティリスト

実行間隔制御部品では、次のプロパティを使用できます。

プロパティキー	プロパティ名	説明	入出力タイプ
interval※	待ち時間	このプロパティには、次のステップを実行するまで待機する時間を 1～1,440（分単位）の範囲で指定します。 デフォルト値は 10 分です。	入力

注※　必須プロパティです。

B.11 戻り値判定分岐部品

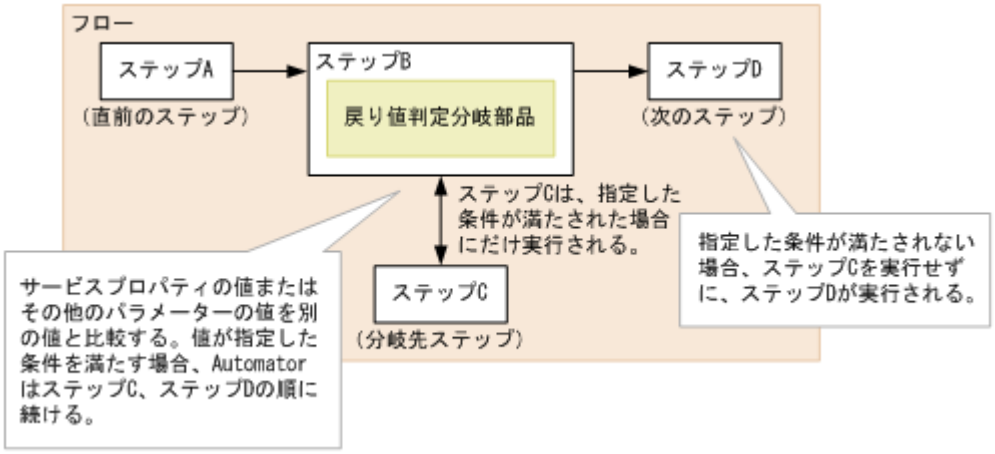
戻り値判定分岐部品は、以前のステップの戻り値をベースとして処理の流れを分岐させます。

この部品により、直前のステップの終了コードに基づいてどのステップを次に実行するかを選択できます。

戻り値判定分岐部品は 2 つの分岐先ステップに接続します。次に続くステップ、および判定条件が満たされる場合にだけ実行されるステップです。終了コードが指定された条件に合致する場合には、Ops Center Automator は分岐先のステップ、および次に続くステップの順で実行します。終了コードが指定した条件に一致しない場合には、Ops Center Automator は次に続くステップのみを実行します。

この部品を値判定部品と一緒に使用すると、文字列に基づいて流れの中のステップを選択できます。

次の図に戻り値判定分岐部品の使用方法を示します。



注意事項

- タスクの実行が部品の実行中に停止するかまたは強制的に終了する場合には、タスクは値判定分岐部品の処理終了後に「正常終了」状態に入ります。
- サービスの実行時にプロパティ値を変更することはできません。フローの作成時に値を設定します。
- 入力プロパティ中の文字列だけを指定できます。サービスプロパティや予約済みプロパティの値をマッピングすることはできません。
- 戻り値判定分岐部品の処理が停止する場合には、hcmds64getlogs コマンドを使ってログ情報を取得して問題を識別します。

終了コード

戻り値判定分岐部品では、次の終了コードが生成されます。

終了コード	説明
0 または 1 以上	部品は正常に終了しました。戻り値判定分岐部品の直前のステップの終了コードがこの部品の終了コードとして設定されます。

プロパティリスト

戻り値判定分岐部品では、次のプロパティを使用できます。

プロパティキー	プロパティ名	説明	入出力タイプ
condition※	判定条件	直前のステップの終了コード用の判定条件を指定します。次の条件から選べます。 ・ ReturnCode=value1 終了コードが Value1 と等しい。 ・ ReturnCode!=value1 終了コードが Value1 と等しくない。 ・ ReturnCode<value1 終了コードが Value1 より小さい。 ・ ReturnCode>value1 終了コードが Value1 より大きい。	入力

プロパティキー	プロパティ名	説明	入出力タイプ
		• ReturnCode<=value1 終了コードが Value1 より小さいかまたは等しい。 • ReturnCode>=value1 終了コードが Value1 より大きいかまたは等しい。 • ReturnCode>value1 AND ReturnCode<value2 終了コードが Value1 より大きくて Value2 より小さい。 • ReturnCode>=value1 AND ReturnCode<value2 終了コードが Value1 より大きいか等しくて Value2 より小さい。 • ReturnCode>value1 AND ReturnCode<=value2 終了コードが Value1 より大きくて Value2 より小さいか等しい。 • ReturnCode>=value1 AND ReturnCode<=value2 終了コードが Value1 より大きいか等しくて Value2 より小さいか等しい。 • ReturnCode<value1 OR ReturnCode>value2 終了コードが Value1 より小さくて Value2 より大きい。 • ReturnCode<=value1 OR ReturnCode>value2 終了コードが Value1 より小さいか等しくて Value2 より大きい。 • ReturnCode<value1 OR ReturnCode>=value2 終了コードが Value1 より小さくて Value2 より大きいか等しい。 • ReturnCode<=value1 OR ReturnCode>=value2 終了コードが Value1 より小さいか等しくて Value2 より大きいか等しい。 デフォルト値は ReturnCode=value1 です。	
value1※	判定条件値 1	終了コードを判定するための数値を 0 から 999 の範囲で指定します。この値は、condition プロパティの value1 にマッピングされます。 デフォルト値は 0 です。	入力
value2	判定条件値 2	終了コードを判定するための数値を 0 から 999 の範囲で指定します。この値は、condition プロパティの value2 にマッピングされます。この値は、条件プロパティ中に Value2 が含まれていると有効になります。 デフォルト値は 0 です。	入力

注※　必須プロパティです。

プロパティの指定例

戻り値判定分岐部品は、終了コードが指定範囲内の値かどうかを決定します。

例として、condition、value1、および value2 プロパティに次の値を使用した場合の有効な判定値の範囲について、以下に説明します。

A. 終了コードは 25 またはそれ以上で 75 より小さい

condition (Condition):ReturnCode>=value1 AND ReturnCode<value2

value1 (Value1):25

value2 (Value2):75

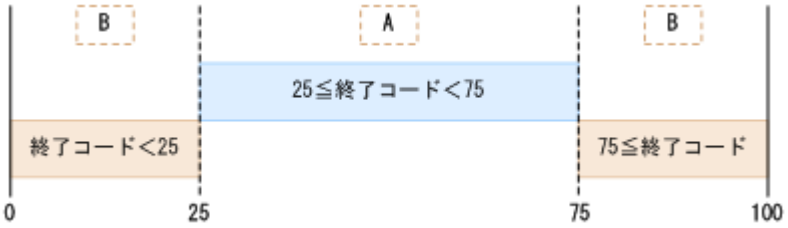
B. 終了コードは 25 より小さいか、または 75 以上

condition (Condition):ReturnCode<value1 OR ReturnCode>=value2

value1 (Value1):25

value2 (Value2):75

それぞれの判定条件に一致する終了コードの範囲を、次の図に示します。



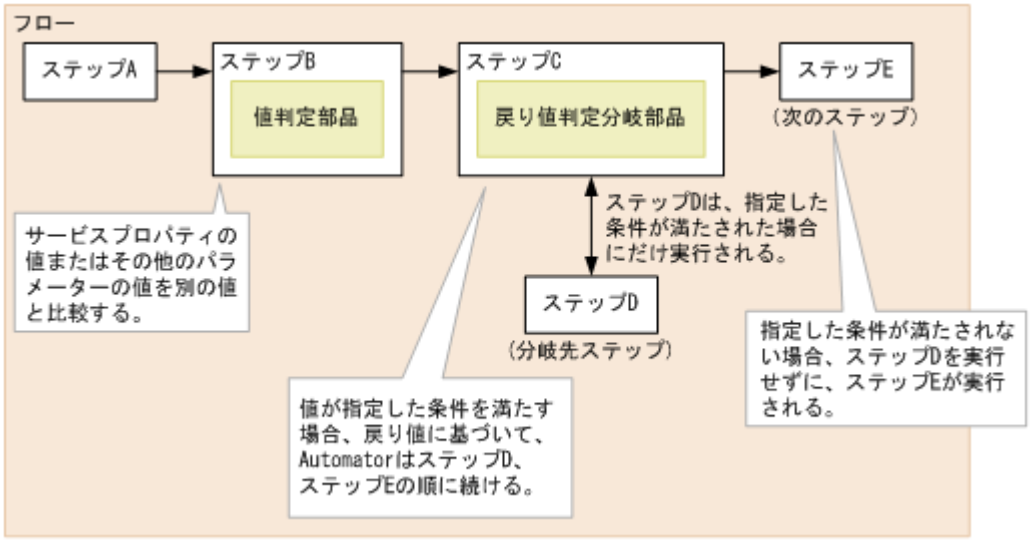
B.12 値判定部品

値判定部品は、サービスのプロパティの値を比較し、値が判定条件と一致する場合は 0 を返します。

この部品は、サービスのプロパティの値、予約プロパティの値、リテラル文字列、またはこれらの任意の組み合わせと、指定された値とを比較します。判定条件が満たされる場合、部品は 0 を返します。

この部品を戻り値判定分岐部品と組み合わせて使用すると、文字列に基づいてフロー内のステップを選択できます。

値判定部品の使用法を、次の図に示します。



注意事項

- 部品の実行中にタスクの実行が停止した場合、値判定部品の処理完了後に、タスクの状態は「正常終了」になります。

終了コード

値判定部品では、次の終了コードが生成されます。

終了コード	説明
0	値が判定条件と一致しました。または、defaultReturnCode プロパティに 0 が指定されています。
1	値が判定条件と一致しませんでした。または、defaultReturnCode プロパティに 1 が指定されています。
63	判定が失敗しました。defaultReturnCode プロパティに 63 が指定されています。
65	Ops Center Automator サーバとの接続に失敗しました。例えば、部品の実行時に Ops Center Automator サーバが停止した可能性があります。
66	次のユーザーが Ops Center Automator ユーザーにマッピングされています。 • Administrators group に属さないユーザー。 • UAC が有効な環境で、Administrators group に属するビルトイン Administrator 以外のユーザー。
68	対象のジョブ実行 ID についての情報がありません。
69	タスク処理エンジンの環境変数を取得できません。
80	タスクの実行が停止しました。
81	部品が不正な状態で呼び出されました。

終了コード	説明
82	タスク処理エンジンからのリクエストメッセージを正しく解析できません。
83	Ops Center Automator サーバの環境が破損しています。
84	指定された部品についての情報を取得できません。
86	指定されたプロパティの値が無効です。
127	そのほかのエラーが発生しました。

プロパティリスト

値判定部品では、次のプロパティを使用できます。

プロパティキー	プロパティ名	説明	入出力タイプ
condition※	判定条件	valueX プロパティの判定条件を指定します。次の条件から選択できます。 ・ valueX=value1 ValueX と Value1 が等しい（数値比較）。 ・ valueX!=value1 ValueX と Value1 が等しくない（数値比較）。 ・ valueX<value1 ValueX が Value1 より小さい（数値比較）。 ・ valueX>value1 ValueX が Value1 より大きい（数値比較）。 ・ valueX<=value1 ValueX が Value1 と等しいか、より小さい（数値比較）。 ・ valueX>=value1 ValueX は Value1 より大きいか等しい（数値比較）。 ・ valueX>value1 AND valueX<value2 ValueX が Value1 より大きく、Value2 より小さい（数値比較）。 ・ valueX>=value1 AND valueX<value2 ValueX は Value1 より大きいか等しくて、Value2 より小さい（数値比較）。 ・ valueX>value1 AND valueX<=value2 ValueX は Value1 より大きくて、Value2 より小さいか等しい（数値比較）。 ・ valueX>=value1 AND valueX<=value2 ValueX は Value1 より大きいか等しくて、Value2 より小さいか等しい（数値比較）。 ・ valueX<value1 OR valueX>value2 ValueX が Value1 より小さいか、Value2 より大きい（数値比較）。 ・ valueX<=value1 OR valueX>value2 ValueX は Value1 より小さいか等しい、または Value2 より大きい（数値比較）。	入力

プロパティキー	プロパティ名	説明	入出力タイプ
		• valueX<value1 OR valueX>=value2 ValueX は Value1 より小さいか、または Value2 より大きいか等しい（数値比較）。 • valueX<=value1 OR valueX>=value2 ValueX は Value1 より小さいか等しい、または Value2 より大きいか等しい（数値比較）。 • valueX equals value1 ValueX と Value1 は等しい。これらの値では大文字と小文字が区別されます（文字列比較）。 • valueX not equals value1 ValueX と Value1 は等しくない。これらの値では大文字と小文字が区別されます（文字列比較）。 • valueX contains value1 ValueX は Value1 を含みます。これらの値では大文字と小文字が区別されます（文字列比較）。 • valueX not contains value1 ValueX は Value1 を含みません。これらの値では大文字と小文字が区別されます（文字列比較）。 デフォルト値は valueX=value1 です。	
valueX※	入力値	比較の基礎となる値を、最大 1,024 文字で指定します。次の形式を単独で、または組み合わせて使用できます。 • ?dna_service-property-key?（サービスプロパティの値を参照している場合） • ?dna_reserved-property-key?（予約プロパティの値を参照している場合） • リテラル文字列	入力
value1	判定条件値 1	valueX プロパティと比較する値を、最大 1,024 文字で指定します。次の形式を単独で、または組み合わせて使用できます。 • ?dna_service-property-key?（サービスプロパティの値を参照している場合） • ?dna_reserved-property-key?（予約プロパティの値を参照している場合） • リテラル文字列 この値は、condition プロパティの value1 にマッピングされます。	入力
value2	判定条件値 2	valueX プロパティと比較する値を、最大 1,024 文字で指定します。次の形式を単独で、または組み合わせて使用できます。 • ?dna_service-property-key?（サービスプロパティの値を参照している場合） • ?dna_reserved-property-key?（予約プロパティの値を参照している場合）	入力

プロパティキー	プロパティ名	説明	入出力タイプ
		・　リテラル文字列 この値は、condition プロパティの value2 にマッピングされます。 このプロパティの値は、condition プロパティに value2 が指定されているとき有効になります。	
defaultReturnCode※	判定失敗時の戻り値	このプロパティは、condition プロパティに数値比較が指定されており、valueX、value1、および value2 プロパティのいずれかに指定された値が数値として比較できない場合、部品により返される値を指定します。 ・ **0** 「値が判定条件に一致する」を判定結果として使用する場合、0 を指定します。 ・ **1** 「値が判定条件に一致しなかった。」を判定結果として使用する場合に 1 を指定します。 ・ **63** 「判定が失敗した」を判定結果として使用し、ステップを異常終了する場合、63 を指定します。 デフォルト値は 63 です。	入力

注※　必須プロパティです。

プロパティ仕様の例

値判定部品は、入力値が指定された値の範囲内かどうかを判定します。

例として、condition、value1、および value2 プロパティに次の値を使用した場合の有効な判定値の範囲について、以下に説明します。

A. 入力値が 25 以上で、75 未満

condition (Condition): ReturnCode>=value1 AND ReturnCode<value2

value1 (Value1): 25

value2 (Value2): 75

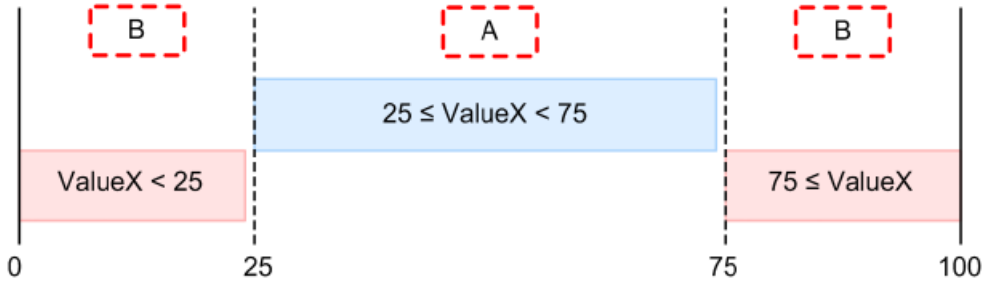
B. 入力値が 25 未満、または 75 以上

condition (Condition): ReturnCode<value1 OR ReturnCode>=value2

value1 (Value1): 25

value2 (Value2): 75

それぞれの判定条件に一致する終了コードの範囲を、次の図に示します。



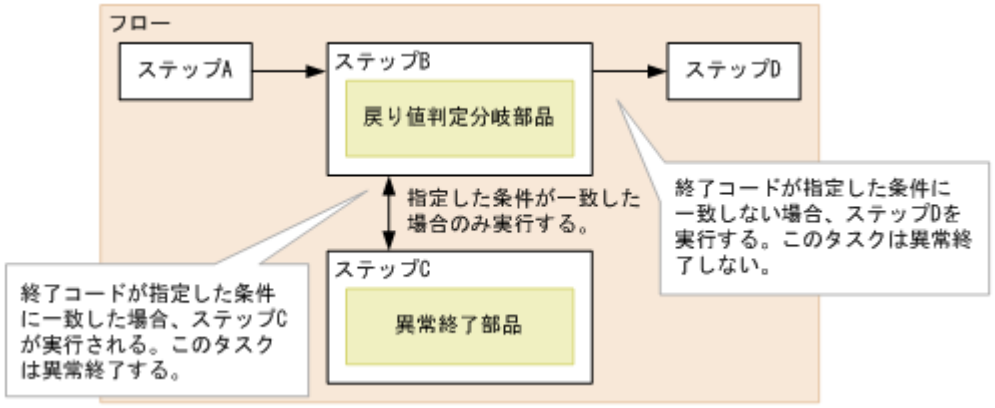
B.13 異常終了部品

異常終了部品は、実行中のフロー、タスク、階層フロー、または繰り返して実行されるフローの異常終了を扱います。

この部品は、実行中のタスクを異常終了させます。

この部品と戻り値判定分岐部品を併用すると、判定条件が合致する場合のフローも異常終了させます。

次の図に、異常終了部品の使用方法を示します。



注意事項

- 部品の実行中にタスクの実行が停止する場合には、異常終了部品の処理終了後にタスクは「異常終了」状態になります。
- 階層フロー部品内で異常終了部品を使用すると、階層フロー部品を扱う階層フローと上位レベルのフローも異常終了します。実行中のタスクも異常終了し、階層フロー部品の戻り値は 0 になります。
- 繰り返し実行部品の中で異常終了部品を使う場合には、繰り返される処理が一度でも異常終了すると繰り返し実行部品の戻り値は 1 になります。繰り返される処理のすべてのインスタンスが異常終了した場合、繰り返し実行部品から戻り値として 2 が返されます。

終了コード

異常終了部品は以下の終了コードを生成します。

終了コード	説明
0	部品は正常に終了しました（ステップは異常終了）。

終了コード	説明
80	タスクの実行が停止しました。

B.14 値判定分岐部品

値判定分岐部品は、サービスのプロパティ値に基づいて処理の流れを分岐します。

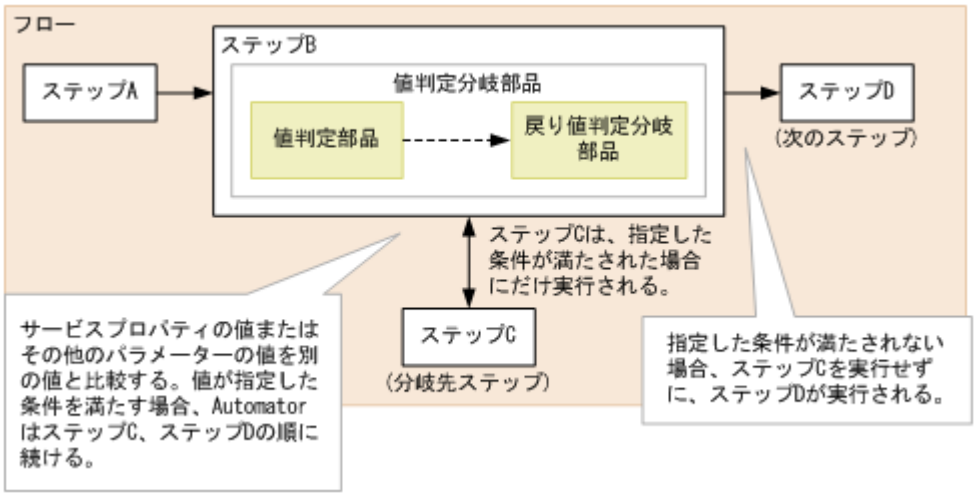
この部品は、サービスプロパティの値、予約プロパティの値、文字列、またはこれらの組み合わせを指定された値と比較します。比較の結果により、Ops Center Automator が次にどのステップを実行するかが決まります。

この値判定分岐部品により、直前のステップの終了コードに基づいてどのステップを次に実行するかを選択できます。

この部品は 2 つの分岐先ステップに接続します。次に続くステップ、および判定条件が満たされる場合にだけ実行されるステップです。入力値が判定条件に合致する場合には、Ops Center Automator は分岐先のステップ、および次に続くステップの順で実行します。入力値が判定条件に一致しない場合には、Ops Center Automator は次に続くステップのみを実行します。

この部品は値判定部品および戻り値判定分岐部品の機能を組み合わせています。

次の図に値判定分岐部品の使用方法を示します。



注意事項

- この部品を実行する際には、タスクログへの情報出力は値判定部品の実行を反映しています。値判定分岐部品はタスクログには貢献しません。
- タスクの実行が部品の実行中に停止する場合には、タスクは値判定分岐部品の処理終了後に「正常終了」状態に入ります。

終了コード

値判定分岐部品では、次の終了コードが生成されます。

終了コード	説明
0	部品は次の場合に 0 を戻します。 ・ 判定結果が真。 ・ 条件プロパティで数値比較が指定されていて、数値ベースでは比較できない値が valueX、value1、および value2 プロパティのどれかで指定されていて、defaultReturnCode プロパティで 0 が指定されています。
1	部品は次の場合に 1 を戻します。 ・ 判定結果が偽。 ・ 条件プロパティで数値比較が指定されていて、数値ベースでは比較できない値が valueX、value1、および value2 プロパティのどれかで指定されていて、defaultReturnCode プロパティで 1 が指定されています。
80	タスクの実行が停止した場合、部品は 80 を戻します。

プロパティリスト

値判定分岐部品では、次のプロパティを使用できます。

プロパティキー	プロパティ名	説明	入出力タイプ
condition※	判定条件	valueX プロパティの分岐条件を指定します。次の条件から選択できます。 ・ valueX>=value1 ValueX は Value1 より大きいか等しい（数値比較）。 ・ valueX>value1 AND valueX<value2 ValueX は Value1 より大きくて Value2 より小さい（数値比較）。 ・ valueX>=value1 AND valueX<value2 ValueX は Value1 より大きいか等しくて、Value2 より小さい（数値比較）。 ・ valueX>value1 AND valueX<=value2 ValueX は Value1 より大きくて、Value2 より小さいか等しい（数値比較）。 ・ valueX>=value1 AND valueX<=value2 ValueX は Value1 より大きいか等しくて、Value2 より小さいか等しい（数値比較）。 ・ valueX<value1 OR valueX>value2 ValueX は Value1 より小さいか、または Value2 より大きい（数値比較）。 ・ valueX<=value1 OR valueX>value2 ValueX は Value1 より小さいか等しい、または Value2 より大きい（数値比較）。 ・ valueX<value1 OR valueX>=value2 ValueX は Value1 より小さいか、または Value2 より大きいか等しい（数値比較）。 ・ valueX<=value1 OR valueX>=value2	入力

プロパティキー	プロパティ名	説明	入出力タイプ
		ValueX は Value1 より小さいか等しい、または Value2 より大きいか等しい（数値比較）。 ・ valueX equals value1 ValueX と Value1 は等しい。この判定は大文字と小文字を区別します（文字列比較）。 ・ valueX not equals value1 ValueX と Value1 は等しくない。この判定は大文字と小文字を区別します（文字列比較）。 ・ valueX contains value1 ValueX は Value1 を含みます。この判定は大文字と小文字を区別します（文字列比較）。 ・ valueX not contains value1 ValueX は Value1 を含みません。この判定は大文字と小文字を区別します（文字列比較）。 デフォルト値は valueX=value1 です。	
valueX※	入力値	比較の基礎となる値を、最大 1,024 文字で指定します。次の形式を単独で、または組み合わせて使用できます。 ・ ?dna_service-property-key?（サービスプロパティの値を参照している場合） ・ ?dna_reserved-property-key?（予約プロパティの値を参照している場合） ・ リテラル文字列	入力
value1	判定条件値 1	valueX プロパティと比較する値を、最大 1,024 文字で指定します。次の形式を単独で、または組み合わせて使用できます。 ・ ?dna_service-property-key?（サービスプロパティの値を参照している場合） ・ ?dna_reserved-property-key?（予約プロパティの値を参照している場合） ・ リテラル文字列 この値は、condition プロパティの value1 にマッピングされます。このプロパティの値は、condition プロパティ中で value1 が指定されると有効になります。	入力
value2	判定条件値 2	valueX プロパティと比較する値を、最大 1,024 文字で指定します。次の形式を単独で、または組み合わせて使用できます。 ・ ?dna_service-property-key?（サービスプロパティの値を参照している場合） ・ ?dna_reserved-property-key?（予約プロパティの値を参照している場合） ・ リテラル文字列 この値は、condition プロパティの value2 にマッピングされます。 このプロパティの値は、condition プロパティ中で value2 が指定されると有効になります。	入力

プロパティキー	プロパティ名	説明	入出力タイプ
defaultReturnCode※	判定失敗時の戻り値	このプロパティは、condition プロパティに数値比較が指定されており、valueX、value1、および value2 プロパティのいずれかに指定された値が数値として比較できない場合、部品により返される値を指定します。 ・ **0** 「値が判定条件に一致する」を判定結果として使用する場合、0 を指定します。 ・ **1** 「値が判定条件に一致しなかった。」を判定結果として使用する場合に 1 を指定します。 ・ **63** 判定結果が Failed でステップを異常終了させたい場合に 63 を指定します。 部品は分岐先のステップを実行することも、続きのステップを実行することもなく異常終了します。 デフォルト値は 63 です。	入力

注※　必須プロパティです。

B.15 ファイルエクスポート部品

指定したファイルに入力コンテンツをエクスポートします。

ファイルエクスポート部品では、あらゆる形式の入力値をあらゆるファイルに出力できます。Apache Velocity Engine VTL（Velocity テンプレート言語）http://velocity.apache.org/を使用して、出力形式を指定できます。形式を指定しない場合、入力値は書式化されずにそのまま出力されます。

ファイルエクスポート部品では、出力ファイルへのフルパスを指定してデータをエクスポートします。また、必要に応じてエラーメッセージが生成されます。

実行の前提条件

- エクスポートするファイルの名前と場所。
- 出力形式に合わせて準備したテンプレート（必要な場合）。

注意事項

- ファイルと同じ名前の出力ファイルがすでに存在する場合、既存の出力ファイルが上書きされます。
- フォルダと同じ名前の出力ファイルがすでに存在する場合、書き込みエラーが発生します。
- エクスポートコンテンツおよび出力テンプレート内にある行送りコードは出力ファイルに含まれます。

終了コード

ファイルエクスポート部品では、次の終了コードが生成されます。

終了コード	説明
0	正常に終了しました。
1	出力ファイルのパスの長さが 256 文字を超えています。
2	VTL 記述内で文法エラーが検出されました。
3	VTL 記述内で未定義の属性またはメソッドが検出された場合、この値が返されます。通常、Velocity が strict モードで実行されていないかぎり、この値は返されません。デフォルトでは、strict モードは false に設定されています。
4	Velocity で使用するリソースファイルが見つかりません。ほとんどの場合、この値は返されません。ファイルエクスポート部品は、リソースファイルを使用しないためです。この値が含まれるのは、Velocity の API によって生成される例外に対応するためです。
5	出力ファイルの書き込み中にエラーが発生しました。
63	部品の処理中にエラーが発生しました。
80	タスクの実行が停止しました。

プロパティリスト

ファイルエクスポート部品では、次のプロパティを使用できます。

プロパティキー	プロパティ名	説明	入出力タイプ
content※	エクスポートコンテンツ	ファイルにエクスポートするコンテンツを指定します。	入力
fileName※	出力ファイル名	コンテンツのエクスポート先であるファイルの名前を指定します。空の文字列は無効です。出力フォルダパスと出力ファイル名で構成されるフルパスの長さが 256 文字を超えることはできません。	入力
directoryPath	出力先ディレクトリパス	出力ファイルの作成先であるフォルダパス。絶対パスまたは相対パスを指定できます。相対パスを指定した場合、タスクの作業フォルダ（Automation/data/task/[tasked]）がパスの開始点とみなされます。フォルダパスを指定しない場合、タスクの作業フォルダ（Automation/data/task/[tasked]）が使用されます。出力フォルダパスと出力ファイル名で構成されるフルパスの長さが 256 文字を超えることはできません。	入力
template	出力用テンプレート	出力コンテンツの形式を定義する VTL 記述が含まれるテンプレートを指定します。	入力
charset	文字セット	出力ファイルの文字セットを指定します。	入力
exportFilePath	出力ファイルパス	エクスポートされる出力ファイルのフルパスを示します。	出力
message	メッセージ	部品の処理中に生成されたメッセージのログを示します。	出力

注※　必須プロパティです。

出力テンプレートでの入力値

入力値にはあらゆる文字を指定できます。入力値が JSON の場合、値は JSON 文字列としてデコードされ、出力テンプレートの VTL 記述の変数 $root にオブジェクトとして格納されます。これにより、オブジェクトを表示できます。入力値が JSON ではない場合、$root に入力値が文字列として保持されます。出力テンプレートの記述方法は、VTL 構文に準拠します。上記の変数 $root は予約済み変数であり、ファイルエクスポート部品がこの変数に値をあらかじめ指定します。また、ファイルエクスポート部品は、CSV 出力用のユーティリティとして予約済み変数 $csv を提供します。次の表に、ファイルエクスポート部品の VTL 記述で使用できる予約済み変数を示します。

変数名	説明
$root	入力値が JSON の場合、値は JSON 文字列としてデコードされ、オブジェクトとして格納されます。入力値が JSON ではない場合、値は文字列として格納されます。
$csv CSV 出力に役立つ以下の方法を提供します。Velocity の ToolManager を使用して、これらを VTL 記述で使用できます。	$csv.value(String) -- 1 つの文字列を 1 つの引数として取得します。CSV 内でエスケープを必要とする文字（二重引用符、コンマ、行送りなど）が含まれる場合、その値を二重引用符で囲みます。文字列値に二重引用符が含まれる場合、その文字列内で二重引用符をエスケープする（" --> ""）ことで、その文字列が出力されます。（CSV 形式の単一セルとしての出力処理）
	$csv.values(String...) -- 複数の文字列を複数の引数（variable-length 引数）として取得し、それらの文字列を CSV 形式の単一行に出力します。単一セルの出力形式は上記の value(String)関数と同じです。
	$csv.values(Collection<String>) -- 配列を取得し、その配列を CSV 形式の単一行に出力します。単一セルの出力形式は上記の value(String)関数と同じです。

B.16 JavaScript 実行部品

JavaScript 実行部品は、指定された JavaScript コードを実行します。この部品は、以下のような特定の JavaScript コードを実行できます。

- 任意のサービスプロパティおよび部品プロパティの値を表示する。
- 部品プロパティを経由して最大 10 個の引数を JavaScript コードに渡す。
- 最大 10 個の値を部品出力プロパティに保存する。
- 値を戻し、戻り値を出力プロパティに保存する。

メモ　JavaScript Plug-in for Configuration Manager REST API は将来のバージョンでサポート終了予定のため、JavaScript 実行部品の auto util ライブラリを使用して Configuration Manager REST API を実行することを推奨します。なお、JavaScript Plug-in for Configuration Manager REST API と同等の API 実行を行いたい場合は、auto util ライブラリの http.handleCall メソッドの 1 番目の引数に auto.util.storage.restCall を指定し、6 番目の引数に auto.util.storage.retrySettings を指定します。詳細はこのトピックの「表 44 auto util メソッド」を参照してください。また、このトピックの「サンプルコード (JavaScript Plug-in for Configuration Manager REST API からの移行)」で、JavaScript Plug-in for Configuration Manager REST API から JavaScript 実行部品に移行する場合のサンプルコードを参照できます。

終了コード

JavaScript 実行部品では、次の終了コードが生成されます。

終了コード	説明
0	正常に終了しました。
1	エラーのためスクリプトが終了しました。
60	JavaScript ライブラリの読み取りエラーが発生しました。
61	JavaScript のコンパイルエラーが発生しました。
62	JavaScript コードの形式が不適切です。
63	部品の処理中にエラーが発生しました。
80	タスクの実行が停止しました。

プロパティリスト

JavaScript 実行部品では、次のプロパティを使用できます。

プロパティキー	プロパティ名	説明	デフォルト値	入出力タイプ	必須
scriptBody	スクリプト本体	JavaScript コードの文字列を指定します。	--	入力	必須
importedScript	インポートスクリプト	同じサービステンプレートに配置されたほかの JavaScript 実行部品と共通で使用されるメソッドと定数(JavaScript のコード文字列)を指定します。 importedScript のスクリプトでは、次の項目を使用できます。 ・ JS ライブラリ : scriptBody で使用できる JS ライブラリと同じです。 ・ 関数 : print()関数。scriptBody で使用できる print()関数と同じです。	--	入力	任意
arg0	スクリプト引数(0)	スクリプトに渡す引数を指定します。	--	入力	任意
arg1	スクリプト引数(1)	スクリプトに渡す引数を指定します。	--	入力	任意
arg2	スクリプト引数(2)	スクリプトに渡す引数を指定します。	--	入力	任意
arg3	スクリプト引数(3)	スクリプトに渡す引数を指定します。	--	入力	任意
arg4	スクリプト引数(4)	スクリプトに渡す引数を指定します。	--	入力	任意
arg5	スクリプト引数(5)	スクリプトに渡す引数を指定します。	--	入力	任意
arg6	スクリプト引数(6)	スクリプトに渡す引数を指定します。	--	入力	任意
arg7	スクリプト引数(7)	スクリプトに渡す引数を指定します。	--	入力	任意

プロパティキー	プロパティ名	説明	デフォルト値	入出力タイプ	必須
arg8	スクリプト引数(8)	スクリプトに渡す引数を指定します。	--	入力	任意
arg9	スクリプト引数(9)	スクリプトに渡す引数を指定します。	--	入力	任意
notify	通知フラグ	通知することがスクリプトで検出された場合に空以外の文字列を指定します。このフラグに空以外の文字列が指定されている場合、部品は戻り値「1」で終了します。	--	出力	任意
returnValue	スクリプト戻り値	指定したスクリプトの機能から返されたオブジェクトの内容が文字列として出力されます。	--	出力	任意
out0	スクリプト出力(0)	ユーザーが指定したスクリプトの2番目の引数のマップ内にあるout0キーに設定された値が出力されます。	--	出力	任意
out1	スクリプト出力(1)	ユーザーが指定したスクリプトの2番目の引数のマップ内にあるout1キーに設定された値が出力されます。	--	出力	任意
out2	スクリプト出力(2)	ユーザーが指定したスクリプトの2番目の引数のマップ内にあるout2キーに設定された値が出力されます。	--	出力	任意
out3	スクリプト出力(3)	ユーザーが指定したスクリプトの2番目の引数のマップ内にあるout3キーに設定された値が出力されます。	--	出力	任意
out4	スクリプト出力(4)	ユーザーが指定したスクリプトの2番目の引数のマップ内にあるout4キーに設定された値が出力されます。	--	出力	任意
out5	スクリプト出力(5)	ユーザーが指定したスクリプトの2番目の引数のマップ内にあるout5キーに設定された値が出力されます。	--	出力	任意
out6	スクリプト出力(6)	ユーザーが指定したスクリプトの2番目の引数のマップ内にあるout6キーに設定された値が出力されます。	--	出力	任意
out7	スクリプト出力(7)	ユーザーが指定したスクリプトの2番目の引数のマップ内にあるout7キーに設定された値が出力されます。	--	出力	任意
out8	スクリプト出力(8)	ユーザーが指定したスクリプトの2番目の引数のマップ内にあるout8キーに設定された値が出力されます。	--	出力	任意

プロパティキー	プロパティ名	説明	デフォルト値	入出力タイプ	必須
out9	スクリプト出力(9)	ユーザーが指定したスクリプトの2番目の引数のマップ内にあるout9キーに設定された値が出力されます。	--	出力	任意

プロパティリストで部品の入力／出力プロパティを指定します。入力プロパティには、サービスプロパティの値、予約プロパティの値、およびリテラル文字の組み合わせを使用できます。

JavaScript 本文に指定できる JavaScript コード

次の表で、スクリプトの本文で使用できる JavaScript コードについて説明します。

表 43 JavaScript コード

<table>
<tr><td colspan="3">文字コード</td><td colspan="3">UTF-8</td></tr>
<tr><td colspan="3">利用可能な JS ライブラリ</td><td colspan="3">underscore.js 1.8.3
auto_util-1.0.1.js（Ops Center Automator のバンドルライブラリ）</td></tr>
<tr><td colspan="3">形式</td><td colspan="3">無名関数である必要があります（サンプルコードを参照）。</td></tr>
<tr><td rowspan="4">関数呼び出しインタフェース</td><td rowspan="4">引数</td><td>serviceProperties</td><td>オブジェクトタイプ</td><td colspan="2">サービスプロパティをマップします。サービスプロパティの値はスクリプトから表示できます。スクリプトで値の更新、削除、追加が行われても、スクリプトの呼び出し後に変更された値はサービスプロパティに反映されません。</td></tr>
<tr><td rowspan="3">pluginProperties</td><td rowspan="3">オブジェクトタイプ</td><td colspan="2">部品プロパティをマップします。部品プロパティの値はスクリプトから表示できます。</td></tr>
<tr><td>arg0～arg9</td><td>部品プロパティに指定した文字列が直接マップされます。文字列が JSON に準拠していても、スクリプトがここを参照した場合は、引数をオブジェクトとしてではなく文字列として取得できます。JSON 文字列から後で説明するオブジェクトへの検証が想定されない場合は、ここを確認します。</td></tr>
<tr><td>notify</td><td>スクリプト内で通知用として指定されたメンバ</td></tr>
</table>

					ーに空文字列以外の値を指定した場合、部品は呼び出し後に戻り値「1」で終了します。
				out0～out9	スクリプト内でout0～out9用として指定されたメンバーに値を指定した場合、値は呼び出し後に部品出力プロパティの引数（0）～引数（9）に反映されます。値はスクリプトの処理結果に反映され、その処理結果を次のステップで使用します（サンプルコードを参照）。
		arg0～arg9	部品プロパティ「引数（0）」～「引数（9）」をマップします（オプション）。 JSON 文字列を部品プロパティに指定した場合、スクリプトの機能内でその文字列をオブジェクトとして取得できます。 ・ 二重引用符で囲まれた文字列はそのままマップされます。 ・ 文字列を囲まない場合、JSON 検証に失敗した文字列はそのままマップされます。		
	終了コード	スクリプトからあらゆるオブジェクトを返すことができます。スクリプトを呼び出した後、終了コードが JSON 文字列として抽出され、その後、JSON 文字列は部品出力プロパティ「戻り値」に反映されます。値はスクリプトの処理結果に反映され、次のステップで使用されます（サンプルコードを参照）。			
print ()関数	スクリプト内で print()関数を使用すると、任意の文字列をタスクログに出力できます。この場合、特定のプレフィックスを文字列の先頭に追加することで、ログレベルを選択します。英字のプレフィックスでは、大文字小文字が区別されます（サンプルコードを参照）。				
	プレフィックス	[Severe]	ログレベル 0 として出力します		
		[Information]	ログレベル 10 として出力します		
		[Fine]	ログレベル 20 として出力します		
		[Finer]	ログレベル 30 として出力します		
		[Debug]	ログレベル 40 として出力します		
		（プレフィックスなし）	プレフィックス[Information]と同じ		
その他の機能	スクリプトで例外がスローされた場合、またはスクリプトで予期しない例外が発生した場合、部品が異常終了し、タスクログに例外が表示されます。				

	スクリプトが適切に終了し、スクリプトから返された値またはスクリプトの部品出力プロパティ（out0～out9）に指定された値に null または未定義の値が含まれる場合、それらの値は null または未定義として格納されます。

auto util ライブラリ

JavaScript 実行部品は、次の機能を持つ auto util ライブラリをサポートしています。

- 任意の宛先に http または https リクエストを送信する
- Web サービス接続の設定値を接続情報として参照できる

次の表に、使用できる auto util メソッドを示します。

表 44 auto util メソッド

メソッド名	引数	戻り値	説明
sleep	スリープする時間を数値型で指定します（ミリ秒単位）。	なし	指定した時間スリープします。
parseJson	JSON の文字列表現を文字列型で指定します。	JSON オブジェクト	文字列を JSON オブジェクトに変換します。
stringifyJson	任意の JSON オブジェクトを指定します。	JSON の文字列表現	JSON オブジェクトを文字列に変換します。
base64.encode	Base64 に変換する文字列を指定します。	Base64 にエンコードされた文字列	Base64 にエンコードします。
base64.decode	Base64 の文字列を指定します。	Base64 からデコードされた文字列	Base64 からデコードします。
http.call	リクエストの JSON オブジェクト	レスポンスの JSON オブジェクト	http または https リクエストを実行し、レスポンスを返します。
storage.restCall	リクエストの JSON オブジェクト	レスポンスの JSON オブジェクト	http または https リクエストを実行し、レスポンスを返します。Configuration Manager REST API を実行する場合にはこのメソッドを使用します。http.call メソッドと比較すると、Accept ヘッダー、タイムアウト時間、およびログ出力の仕様が異なります。詳細については、「表 48 auto util ライブラリの http.call と storage.restCall の仕様の違い」を参照してください。
http.toRawHeader	ヘッダーの JSON オブジェクト	ヘッダー文字列	文字列の形式でヘッダーを返します。引数には、キーとその値が設定されている JSON オブジェクトを指定します。
http.defaultErrorHandler	エラー リクエストの JSON オブジェクト レスポンスの JSON オブジェクト	なし	HttpError インスタンスをスローします。

メソッド名	引数		戻り値	説明
http.handleCall	1	httpCall メソッド	なし	2 番目の引数にリクエストを指定することで、1 番目の引数の httpCall メソッドを呼び出します。ステータスコードが 200 以上 400 未満の場合は 3 番目の引数で指定したメソッドを呼び出します。その他のステータスコードの場合は 4 番目の引数で指定したメソッドを呼び出します。http 呼び出しが失敗し、http のレスポンスが受け取れなかった場合は、5 番目の引数で指定したメソッドを呼び出します。 1 番目の引数に httpCall メソッドを指定すると、リクエスト、レスポンス、およびエラーの JSON オブジェクトは、それぞれ変数 req、resp、および err で呼び出され、3～5 番目の関数オブジェクトの引数に使用することができます。 6 番目の引数に auto.util.storage.retrySettings を指定した場合、HTTP レスポンスが次の条件を満たす際に、次の回数と間隔でリトライが実行されます。 • リトライの条件：以下のいずれかを満たす ◦ ステータスコードが 503 ◦ ステータスコードが 500 でレスポンスヘッダーの Content-Type が text/html • リトライ回数：130 回 （外部リソースプロバイダを実行する場合は 30 回） • リトライ間隔：30 秒 （外部リソースプロバイダを実行する場合は 10 秒）
	2	リクエストの JSON オブジェクト		
	3	結果のステータスコードが 200 以上 400 未満の場合に呼び出すメソッド		
	4	結果のステータスコードが 200 以上 400 未満でない場合に呼び出すメソッド		
	5	http 呼び出しが失敗し、http のレスポンスが受け取れなかった場合に呼び出すメソッド		
	6	リトライ条件を持つオブジェクト（このオブジェクトのメンバーについては、「表 47 auto util ライブラリの http.handleCall メソッドの 6 番目の引数メンバー」を参照。）		
http.getHeaderValue	レスポンスの JSON オブジェクトのヘッダーを文字列型で指定		ヘッダーのキー名に対応する値	レスポンスヘッダーの値を取得します。レスポンスヘッダーのキーが重複している場合は、最初に見つかったヘッダーが返されます。 存在しないキー名を指定した場合は、空文字が返却されます。
	取得するヘッダーのキー名を文字列型で指定。指定する値は、大文字・小文字を区別しない。			
http.buildQuery	指定したいクエリパラメタをマップ形式で指定		クエリパラメタを文字列形式で出力	クエリパラメタの構築および URL のエンコードを行います。使用例については、このトピックの「サンプルコード（JavaScript Plug-in for Configuration Manager REST API からの移行）」の JavaScript 実行部品のサンプルコードを参照してください。

メソッド名	引数	戻り値	説明
env.getServiceTemplate	なし	サービステンプレートオブジェクト	サービステンプレートの JSON オブジェクトを取得します。
env.getService	なし	サービスオブジェクト	サービスの JSON オブジェクトを取得します。
env.getWebServiceConnections	Web サービス接続先のカテゴリ名	入力パラメーターとして指定されたカテゴリ名を持つ Web サービス接続先の JSON オブジェクトのリスト	Web サービス接続先の JSON オブジェクトのリストを取得します。サービスまたはタスクによるアクセスが可能な Web サービス接続先が取得されます。その際に、サービス、サービスグループ、インフラストラクチャーグループ、および Web サービス接続先の定義済みの関連性が考慮されます。引数を指定しない場合は、全カテゴリを取得します。

リクエストの JSON オブジェクトには、次のメンバーを使用できます。

表 45 JSON オブジェクト

メンバー	種別	説明
requestUrl	String	［Web サービス接続を使用していない場合］http または https で始まるリクエスト URL を指定します。［Web サービス接続を使用している場合］リクエスト URL のホスト名のあとのスラッシュ（/）から en までの部分を指定します。例：http://host:port/Folder/の場合、「/Folder/」を指定します。
requestMethod	String	HTTP リクエストメソッド。次の値を文字列として指定します。 ・ GET ・ POST ・ PUT ・ DELETE ・ PATCH authScheme に none または basic を指定した場合にだけ PATCH を指定できます。
requestHeaders	String	リクエストヘッダー。http.toRawHeader メソッドの戻り値が設定されているとします。［Web サービス接続を使用している場合］ユーザー ID およびパスワードの埋め込み文字として、次の値を使用できます。- ${connection.username} - ${connection.password}
requestBody	String	リクエストボディ
authScheme	String	次のいずれかの値を必ず指定してください。 ・ none ・ basic ・ digest ・ negotiate JavaScript 実行部品または外部リソースプロバイダの JavaScript コードによってリクエストに

メンバー	種別	説明
		Authorization ヘッダーを指定する場合は、none を指定してください。
productName	String	Web サービス接続カテゴリを指定します。Web サービス接続を使用する場合は、必ず指定してください。
connectionName	String	Web サービス接続名を指定します。Web サービス接続を使用する場合は、必ず指定してください。
userName	String	接続先に対して認証するユーザー名を指定します。Web サービス接続を使用する場合は、指定しないでください。
password	String	接続先に対して認証するパスワードを指定します。
useProxy	boolean	プロキシを使用するかどうかを指定します（true または false）。
proxyHost	String	プロキシサーバのホスト名または IP アドレスを指定します。
proxyPort	int	プロキシサーバのポート番号を指定します。
proxyAuthScheme	String	useProxy に true を指定した場合は、次のいずれかの値を必ず指定してください。 ・ none ・ basic ・ digest
proxyUserName	String	プロキシサーバで認証が必要な場合は、ユーザー名を指定します。
proxyPassword	String	プロキシサーバで認証が必要な場合は、パスワードを指定します。
connectionTimeout	int	HTTP または HTTPS で接続する際にタイムアウト値を秒単位（0～3600）で指定します。0 を指定した場合はタイムアウトしません。値域外の場合はプロパティ値が使用されます。
readTimeout	int	HTTP または HTTPS で接続確立後、データ読み取り時のタイムアウト値を秒単位（0～86400）で指定します。0 を指定した場合はタイムアウトしません。値域外の場合はプロパティ値が使用されます。
outputLog	boolean	レスポンスボディをタスクログに出力するかどうかを指定します。true および false 以外が指定された場合はデフォルト値（true）が使用されます。なお、storage.restCall メソッドを実行する場合だけ指定できます。

次の表に、対応できるレスポンスメンバーを示します。

表 46 レスポンスメンバー

メンバー	種別	説明
responseHeaders	String	レスポンスヘッダー
responseStatusCode	int	レスポンスコード
responseStatusMessage	String	レスポンスメッセージ

メンバー	種別	説明
responseBody	String	レスポンスボディ

auto util ライブラリの http.handleCall メソッドの 6 番目の引数には、次のメンバーを使用できます。

表 47 auto util ライブラリの http.handleCall メソッドの 6 番目の引数メンバー

メンバー	種別	説明
retryCondition	Function object	直前に実行された API のレスポンスを基に、リトライするかどうかを返す関数です。true の場合はリトライを実行し、false の場合はリトライしません。使用例については、このトピックの「サンプルコード（リトライの設定）」を参照してください。
maxRetryCount	int	リトライ回数。0 以上 1000 以下の整数を指定してください。
retryWaitTime	int	リトライ間隔（秒）。0 以上 3600 以下の整数を指定してください。

次の表に、auto util ライブラリの http.call と storage.restCall の仕様の違いを示します。

表 48 auto util ライブラリの http.call と storage.restCall の仕様の違い

項目	http.call	storage.restCall
Accept ヘッダー	Accept：application/json	Accept：*/*
タイムアウト時間	Connection：60 秒（1 分） Read：600 秒（10 分）	Connection：600 秒（10 分） Read：10800 秒（180 分）
ログ出力	リクエスト URL、レスポンスコード、およびレスポンスボディのログはタスクログに出力されません。	リクエスト URL、レスポンスコード、およびレスポンスボディのログがタスクログに出力されます。

サンプルコード（scriptBody）

部品プロパティ「JavaScript Body」に指定できる JavaScript サンプルコードを以下に示します。

```
function(serviceProperties, pluginProperties, arg0, arg1, arg2) {
    var obj = new Object();
    print("[Debug] Function begin.");

    obj.mem1 = arg0;
    obj.mem2 = arg1;

    if (arg2 == "") {
        pluginProperties["notify"] = 999;
        pluginProperties["out1"] = "NOTE!:The arg2 is EMPTY.";
    } else {
        obj.mem3 = arg2;
        obj.status = "success";
        pluginProperties["out1"] = "Finished successfully.";
    }

    print("[Debug] Function end.");

    return obj;
}
```

上記の表で説明しているように、arg0、arg1...を取得するには 2 つの方法があり、それぞれの方法で異なる値が提供されます。pluginProperties の場合、引数は文字列として取得されますが、引数 arg0、arg1...から取得する場合、JavaScript から評価値を取得できます。部品の入力プロパティを pluginProperties から取得した場合と、arg0、arg1...から取得した場合の違いを、次の例に示します。

arg0 に 3 を、arg1 に 5 を指定して次のサンプルコードを実行するとします。

```
function(serviceProperties, pluginProperties, arg0, arg1) {
    pluginProperties["out0"] = pluginProperties["arg0"] +
pluginProperties["arg1"];
    pluginProperties["out1"] = arg0 + arg1;
}
```

このサンプルコードを実行すると、out0 に「35」が、out1 に「8.0」が設定されます。pluginProperties の場合、引数はそれぞれ「3」、「5」の文字列として扱われ、「+」記号は文字列連結として扱われます。また、arg0 および arg1 の場合、それぞれ 3.0 と 5.0 になり、「+」記号は加算演算として扱われます。

以下のサンプルでは、Web サービス接続により提供される接続情報を参照することで、ストレージシステムに含まれる Platform REST API サーバからプール一覧を取得し、出力プロパティ out0 に設定しています。前提条件として、カテゴリが「StorageSystem」で名前が「storage1」の Web サービス接続先がすでに登録されている必要があります。

```
function getPools(productname, connectionname) {
  var respBody = null;
  var request = {
    requestMethod:'GET',
    requestUrl:'/ConfigurationManager/v1/objects/pools',
    requestHeaders:auto.util.http.toRawHeader({
      'Accept':'application/json',
      'Accept-Language':'en',
      'Content-Type':'application/json',
    }),
    authScheme:'basic',
    connectionName: connectionname,
    productName: productname,
  };
  auto.util.http.handleCall(auto.util.http.call, request,
    function(resp, req) {
      respBody = resp.responseBody;
    }, function(resp, req) {
      auto.util.http.defaultErrorHandler(null, req, resp);
    }, function(err, req) {
      auto.util.http.defaultErrorHandler(err, req);
    }
  );
  return JSON.parse(respBody);
}
function fn(serviceProperties, pluginProperties, arg0, arg1, arg2, arg3,
arg4, arg5, arg6, arg7, arg8, arg9) {
  var data = getPools('StorageSystem', 'storage1').data;
  var poolList = [];
  _.each(data, function(d){
    poolList.push({
      poolId: d.poolId,
      poolStatus: d.poolStatus,
      availableVolumeCapacity: d.availableVolumeCapacity
    });
  });
  pluginProperties.out0 = JSON.stringify(poolList, null, 2);
}
```

以下のサンプルでは、Web サービス接続の接続情報を参照せずに、ストレージシステムに含まれる Platform REST API サーバからプール一覧を取得し、出力プロパティ out0 に設定しています。前

提条件として、Platform REST API サーバのホスト名が「host」、ログインユーザー名が「user」、パスワードが「password」である必要があります。

```
function getPools() {
  var respBody = null;
  var request = {
    requestMethod:'GET',
    requestUrl:'https://host/ConfigurationManager/v1/objects/pools',
    requestHeaders:auto.util.http.toRawHeader({
      'Accept':'application/json',
      'Accept-Language':'en',
      'Content-Type':'application/json',
    }),
    authScheme:'basic',
    userName:'user',
    password:'password',
    useProxy:'false'
  };
  auto.util.http.handleCall(auto.util.http.call, request,
    function(resp, req) {
      respBody = resp.responseBody;
    }, function(resp, req) {
      auto.util.http.defaultErrorHandler(null, req, resp);
    }, function(err, req) {
      auto.util.http.defaultErrorHandler(err, req);
    }
  );
  return JSON.parse(respBody);
}
function fn(serviceProperties, pluginProperties, arg0, arg1, arg2, arg3,
arg4, arg5, arg6, arg7, arg8, arg9) {
  var data = getPools().data;
  var poolList = [];
  _.each(data, function(d){
    poolList.push({
      poolId: d.poolId,
      poolStatus: d.poolStatus,
      availableVolumeCapacity: d.availableVolumeCapacity
    });
  });
  pluginProperties.out0 = JSON.stringify(poolList, null, 2);
}
```

サンプルコード（scriptBody／importedScript）

部品プロパティ「scriptBody」「importedScript」に指定できる JavaScript サンプルコードを以下に示します。

```
-----------------------------------------
(scriptBody)
-----------------------------------------
function fn(serviceProperties, pluginProperties, arg0, arg1, arg2, arg3,
arg4, arg5, arg6, arg7, arg8, arg9) {

  hoge(CNST);

}
-----------------------------------------
(importedScript)
-----------------------------------------
var CNST = "hoge";

function hoge(a){
  print(a + " from common js!");
}
-----------------------------------------
```

サンプルコード（リトライの設定）

auto util ライブラリの http.handleCall メソッドの 6 番目の引数としてカスタム条件を指定する場合のサンプルコードを以下に示します。

```
var retrySettings = {};
retrySettings.retryCondition = function (response) {return
response.responseStatusCode===503;};  // レスポンスが 503 の時にリトライする
retrySettings.maxRetryCount = 10;  // 最大 10 回リトライする
retrySettings.retryWaitTime = 60;  // 60 秒毎にリトライする

auto.util.http.handleCall(auto.util.storage.restCall, request,
successHandler, serverErrorHandler, clientErrorHandler, retrySettings);
```

サンプルコード（JavaScript Plug-in for Configuration Manager REST API からの移行）

Configuration Manager REST API を実行するために使用される部品として、JavaScript Plug-in for Configuration Manager REST API から JavaScript 実行部品へ移行する場合のサンプルコードを次に示します。

以下に、移行前の JavaScript Plug-in for Configuration Manager REST API のサンプルコードを示します。

プール情報を取得する

```
function fn(serviceProperties, pluginProperties, arg0, arg1, arg2, arg3,
arg4, arg5, arg6, arg7, arg8, arg9) {
  var storageDeviceId = <storageDeviceId>;

  //Step 1. Instantiate the API Client.
  var client = new api.ObjectsApi();

  //Step 2. Specify the User Credentials. There are two methods for
specifying user credentials. You can use either method, but the first
one is recommended.
  //Specify "Web Service Connection Category" and "Web Service
Connection Name" in the plug-in properties. You do not need to specify
credentials in a script.

  //Step 3. Get Session
  //You can pass arguments by using the plug-in input properties or you
can specify them directly in a script. For example, you can specify arg3
for the device ID.
  var argSessionsPost = new
argDef.ObjectsApi.v1.objects.storages.storageDeviceID.sessions.post();
  argSessionsPost.setStorageDeviceID(storageDeviceId);
  var responseBody =
client.v1.objects.storages.storageDeviceID.sessions.post(argSessionsPost)
;
  var token = responseBody.getToken();
  var sId = responseBody.getSessionId();

  //Step 4. Call the API that is associated with your use case based on
the session obtained in Step 3.
  var argPoolsGet = new
argDef.ObjectsApi.v1.objects.storages.storageDeviceID.pools.get();
  argPoolsGet.setStorageDeviceID(storageDeviceId);
  argPoolsGet.setPoolType("DP");
  client.getApiClient().setApiKeyPrefix("Session");
  client.getApiClient().setApiKey(token);
  var ret =
client.v1.objects.storages.storageDeviceID.pools.get(argPoolsGet);

  //Final Step. Set the required information for the output property of
the plug-in or return value.
  pluginProperties.out0 = ret.getData()[0].getPoolId();
  pluginProperties.out1 = ret.getData()[0].getPoolName();
```

```
  return ret;

}
```

以下に、移行後の JavaScript 実行部品のサンプルコードを示します。

プール情報を取得する

```
function fn(serviceProperties, pluginProperties, arg0, arg1, arg2, arg3,
arg4, arg5, arg6, arg7, arg8, arg9) {
  var storageDeviceId = <storageDeviceId>;

  //Step 1. Generate a method to run REST API to Configuration Manager.
  var configurationManagerCall = function(request) {
    var respBody = null;
    auto.util.http.handleCall(auto.util.storage.restCall, request,
      function(resp, req) {
        respBody = auto.util.parseJson(resp.responseBody);
      }, function(resp, req) {
        auto.util.http.defaultErrorHandler(null, req, resp);
      }, function(err, req) {
        auto.util.http.defaultErrorHandler(err, req);
      }, auto.util.storage.retrySettings
    );
    return respBody;
  };

  //Step 2. Get accessible Web Service Connections by specifying a
category name.
  var wsc =
auto.util.env.getWebServiceConnections("ConfigurationManager");

  //Step 3. Get Session
  //You can pass arguments by using the plug-in input properties or you
can specify them directly in a script. For example, you can pass the
device ID through the query parameters (queryParamMap).
  var request = {
    "requestMethod": "POST",
    "requestUrl": "/ConfigurationManager/v1/objects/storages/" +
storageDeviceId + "/sessions",
    "requestHeaders": auto.util.http.toRawHeader({}),
    "authScheme": "basic",
    "connectionName": wsc[0].name,
    "productName": wsc[0].productName
  };
  var responseBody = configurationManagerCall(request);
  var token = responseBody.token;
  var sId = responseBody.sessionId;

  //Step 4. Call the API that is associated with your use case based on
the session obtained in Step 3.
//When specifying the Authorization header for the request in the user
program, specify "none" for authScheme.
  var headers = {
    "Authorization": "Session " + token,
  };
  var queryMap = {
    "poolType" : "DP"
  };
  var queryStr = auto.util.http.buildQuery(queryMap);
  request = {
    "requestMethod": "GET",
    "requestUrl": "/ConfigurationManager/v1/objects/storages/" +
storageDeviceId + "/pools" + queryStr,
    "requestHeaders": auto.util.http.toRawHeader(headers),
    "authScheme": "none",
    "connectionName": wsc[0].name,
    "productName": wsc[0].productName
  };
  var ret = configurationManagerCall(request);

  //Final Step. Set the required information for the output property of
```

```
the plug-in or return value.
  pluginProperties.out0 = ret.data[0].poolId;
  pluginProperties.out1 = ret.data[0].poolName;
  return ret;
}
```

B.17 JavaScript Plug-in for Configuration Manager REST API

JavaScript Plug-in for Configuration Manager REST API は、JavaScript で書かれたスクリプトを実行し、Configuration Manager REST API にアクセスするためのメソッドを含んでいます。

メモ　JavaScript Plug-in for Configuration Manager REST API は将来のバージョンでサポート終了予定のため、JavaScript 実行部品の auto util ライブラリを使用して Configuration Manager REST API を実行することを推奨します。JavaScript 実行部品の詳細については「B.16 JavaScript 実行部品」を参照してください。

Configuration Manager REST API の指定と、この部品のスクリプトで使用できるメソッドおよびデータモデルの指定との関係は次のとおりです。

- 各 API のパス名、つまり「/」（スラッシュ）区切り文字が「.」(ドット)区切り文字に置換された名前を有するメソッドが、部品を介して提供されます。
- HTTP メソッドは、メソッド名の末尾に、「~.get(...」、「~.post(...」のように指定できます。
- API パスの変数とクエリパラメタは、メソッドの引数として指定できます。
- リクエストボディを関連データモデルとして引数に与えることができます。

Configuration Manager REST API の HTTP ステータスコードが 503 の場合、次のリトライ回数と間隔でリトライが実行されます。

- リトライ回数：130
- リトライ間隔：30 秒

本部品の最大同時実行数は 30 です。config_user.properties の plugin.threadPoolSize に 30 を超える値を指定していても本部品の同時実行数は 30 に制限されます。

plugin.threadPoolSize に 30 以下の値を指定している場合は、本部品の最大同時実行数は plugin.threadPoolSize の指定値となります。

終了コード

JavaScript Plug-in for Configuration Manager REST API では、次の終了コードが生成されます。

終了コード	説明
0	正常に終了しました。
1	エラーのためスクリプトが終了しました。
60	JavaScript ライブラリの読み取りエラーが発生しました。
61	JavaScript のコンパイルエラーが発生しました。
62	JavaScript コードの形式が不適切です。
63	部品の処理中にエラーが発生しました。
80	タスクの実行が停止しました。

プロパティリスト

JavaScript Plug-in for Configuration Manager REST API では、次のプロパティを使用できます。

プロパティキー	プロパティ名	説明	入出力タイプ
webServiceConnectionCategory	Web Service Connection Category	Web サービス接続カテゴリを指定します。このプロパティと Web サービス接続名の組み合わせにより、Web サービス接続を一意に識別できます。	入力
webServiceConnectionName	Web Service Connection Name	Web サービス接続名を指定します。このプロパティと Web サービス接続カテゴリを指定すると、Ops Center Automator に登録されている Web サービス接続が検索され、一致する接続があれば、その URL とプロキシが API の呼び出し時に使用されます。	入力
baseUrl	Base URL	API の呼び出し時のベース URL を指定します。- 入力形式は次のとおりです。＜プロトコル＞://＜ホスト＞:＜ポート＞/- その後に続けてリソースのパスとクエリパラメタを指定することはできません。Web サービス接続の詳細を参照してください。	入力
requestHeaders	Request Headers	API の呼び出しごとに追加される HTTP リクエストヘッダーを指定します。主に、各種認証に使用されます。次の形式で指定します（ヘッダー名と値の先頭と末尾の空白は無視されます）。＜*ヘッダー 1 のヘッダー名*＞:＜*ヘッダー 1 の値*＞¥n＜*ヘッダー 2 のヘッダー名*＞:＜*ヘッダー 2 の値*＞¥n ヘッダーを任意に追加指定できます。ただし、Configuration Manager REST API との接続に必要なヘッダー（認証情報など）は自動的に追加されるため、基本的には、ここでヘッダーを指定する必要はありません。	入力
webUsername	Server Authentication Basic User Name	サーバ認証に使用されるユーザー名を指定します。	入力
webPassword	Server Authentication Basic Password	サーバ認証に使用されるパスワードを指定します。	入力
useProxy	Use Proxy Server	プロキシ接続が必要な場合は true に設定します。	入力
ApiKey	Server Authentication API Key	認証用の API キーを指定します（API キーのプレフィックスと共に使用されます）。	入力
ApiKeyPrefix	Server Authentication API Key Prefix	認証用の API キーのプレフィックスを指定します（API キーと共に使用されます）。HTTP ヘッダーの次の部分です。	入力

プロパティキー	プロパティ名	説明	入出力タイプ
		Authorization:[API Key Prefix] [API Key]	
proxyHostname	Proxy Hostname	プロキシのホスト名または IP アドレスを指定します。	入力
proxyPort	Proxy Port Number	プロキシのポート番号を指定します。	入力
proxyAuth	Proxy Server Authentication Scheme	プロキシ認証の種別を指定します。次の認証機能がサポートされています。 ・ Basic 認証 ・ Digest 認証	入力
proxyUsername	Proxy Username	プロキシ認証に使用されるユーザー名を指定します。	入力
proxyPassword	Proxy Password	プロキシ認証に使用されるパスワードを指定します。	入力
scriptBody※	Script body	JavaScript コードの文字列を指定します。	入力
importedScript	Imported script	同じサービステンプレートに配置されたほかの JavaScript Plug-in for Configuration Manager REST API と共通で使用されるメソッドと定数（JavaScript のコード文字列）を指定します。 importedScript のスクリプトでは、次の項目を使用できます。 ・ JS ライブラリ：scriptBody で使用できる JS ライブラリと同じです。 ・ 関数：print()関数。scriptBody で使用できる print()関数と同じです。	入力
arg0	Argument(0)	スクリプトに渡す引数を指定します。	入力
arg1	Argument(1)	スクリプトに渡す引数を指定します。	入力
arg2	Argument(2)	スクリプトに渡す引数を指定します。	入力
arg3	Argument(3)	スクリプトに渡す引数を指定します。	入力
arg4	Argument(4)	スクリプトに渡す引数を指定します。	入力
arg5	Argument(5)	スクリプトに渡す引数を指定します。	入力
arg6	Argument(6)	スクリプトに渡す引数を指定します。	入力
arg7	Argument(7)	スクリプトに渡す引数を指定します。	入力
arg8	Argument(8)	スクリプトに渡す引数を指定します。	入力
arg9	Argument(9)	スクリプトに渡す引数を指定します。	入力
notify	Notification flag	通知することがスクリプトで検出された場合に空以外の文字列を指定します。このフラグに空以外の文字列が指定された場合、部品は戻り値「1」で終了します。	出力
returnValue	Return value	指定したスクリプトの機能から返されたオブジェクトの内容が文字列として出力されます。	出力

プロパティキー	プロパティ名	説明	入出力タイプ
out0	Output(0)	ユーザーが指定したスクリプトの 2 番目の引数のマップ内にある out0 キーに設定された値が出力されます。	出力
out1	Output(1)	ユーザーが指定したスクリプトの 2 番目の引数のマップ内にある out1 キーに設定された値が出力されます。	出力
out2	Output(2)	ユーザーが指定したスクリプトの 2 番目の引数のマップ内にある out2 キーに設定された値が出力されます。	出力
out3	Output(3)	ユーザーが指定したスクリプトの 2 番目の引数のマップ内にある out3 キーに設定された値が出力されます。	出力
out4	Output(4)	ユーザーが指定したスクリプトの 2 番目の引数のマップ内にある out4 キーに設定された値が出力されます。	出力
out5	Output(5)	ユーザーが指定したスクリプトの 2 番目の引数のマップ内にある out5 キーに設定された値が出力されます。	出力
out6	Output(6)	ユーザーが指定したスクリプトの 2 番目の引数のマップ内にある out6 キーに設定された値が出力されます。	出力
out7	Output(7)	ユーザーが指定したスクリプトの 2 番目の引数のマップ内にある out7 キーに設定された値が出力されます。	出力
out8	Output(8)	ユーザーが指定したスクリプトの 2 番目の引数のマップ内にある out8 キーに設定された値が出力されます。	出力
out9	Output(9)	ユーザーが指定したスクリプトの 2 番目の引数のマップ内にある out9 キーに設定された値が出力されます。	出力

注※　必須プロパティです。

Web サービス接続の設定値を、部品の入力プロパティの値として参照できます。この場合、対応する入力プロパティは Web サービス接続の設定値で上書きされます。Web サービス接続の設定値を参照するかどうかは、webServiceConnectionCategory と webServiceConnectionName の入力値で決まります。Web サービス接続の設定値を参照する場合、URL（プロパティ名：baseUrl）は無視されます。

プロパティリストで部品の入力／出力プロパティを指定します。入力プロパティには、サービスプロパティの値、予約プロパティの値、およびリテラル文字の組み合わせを使用できます。

webServiceConnectionCategory の入力値	webServiceConnectionName の入力値	部品の動作
あり	あり	Web サービス接続の値を参照します。一致する Web サービス接続が見つからない場合、実行時エラーになります。

webServiceConnectionCategory の入力値	webServiceConnectionName の入力値	部品の動作
あり	なし	エラー（Web サービス接続の参照を試みましたが、一致するものが見つかりませんでした。）
なし	あり	エラー（Web サービス接続の参照を試みましたが、一致するものが見つかりませんでした。）
なし	なし	Web サービス接続の値を参照しません。

JavaScript 本文に指定できる JavaScript コード

次の表で、スクリプトの本文で使用できる JavaScript コードについて説明します。

<table>
<tr><td colspan="3">文字コード</td><td colspan="3">UTF-8</td></tr>
<tr><td colspan="3">利用可能な JS ライブラリ</td><td colspan="3">underscore.js 1.8.3
auto_util-1.0.1.js（Ops Center Automator のバンドルライブラリ）</td></tr>
<tr><td colspan="3">形式</td><td colspan="3">無名関数である必要があります（サンプルコードを参照）。</td></tr>
<tr><td rowspan="4">関数呼び出しインタフェース</td><td rowspan="4">引数</td><td>serviceProperties</td><td>オブジェクトタイプ</td><td colspan="2">サービスプロパティをマップします。サービスプロパティの値はスクリプトから表示できます。スクリプトで値の更新、削除、追加が行われても、スクリプトの呼び出し後に変更された値はサービスプロパティに反映されません。</td></tr>
<tr><td rowspan="3">pluginProperties</td><td rowspan="3">オブジェクトタイプ</td><td colspan="2">部品プロパティをマップします。部品プロパティの値はスクリプトから表示できます。</td></tr>
<tr><td>arg0～arg9</td><td>部品プロパティに指定した文字列が直接マップされます。文字列が JSON に準拠していても、スクリプトがここを参照した場合は、引数をオブジェクトとしてではなく文字列として取得できます。JSON 文字列から後で説明するオブジェクトへの検証が想定されない場合は、ここを確認します。</td></tr>
<tr><td>notify</td><td>スクリプト内で通知用として指定されたメンバ</td></tr>
</table>

					ーに空文字列以外の値を指定した場合、部品は呼び出し後に戻り値「1」で終了します。
				out0～out9	スクリプト内でout0～out9用として指定されたメンバーに値を指定した場合、値は呼び出し後に部品出力プロパティの引数（0）～引数（9）に反映されます。値はスクリプトの処理結果に反映され、その処理結果を次のステップで使用します（サンプルコードを参照）。
		arg0～arg9	部品プロパティ「引数（0）」～「引数（9）」をマップします（オプション）。 JSON 文字列を部品プロパティに指定した場合、スクリプトの機能内でその文字列をオブジェクトとして取得できます。 ・ 二重引用符で囲まれた文字列はそのままマップされます。 ・ 文字列を囲まない場合、JSON 検証に失敗した文字列はそのままマップされます。		
	終了コード	スクリプトからあらゆるオブジェクトを返すことができます。スクリプトを呼び出した後、終了コードが JSON 文字列として抽出され、その後、JSON 文字列は部品出力プロパティ「戻り値」に反映されます。値はスクリプトの処理結果に反映され、次のステップで使用されます（サンプルコードを参照）。			
print ()関数	スクリプト内で print()関数を使用すると、任意の文字列をタスクログに出力できます。この場合、特定のプレフィックスを文字列の先頭に追加することで、ログレベルを選択します。英字のプレフィックスでは、大文字小文字が区別されます（サンプルコードを参照）。				
	プレフィックス	[Severe]	ログレベル 0 として出力します		
		[Information]	ログレベル 10 として出力します		
		[Fine]	ログレベル 20 として出力します		
		[Finer]	ログレベル 30 として出力します		
		[Debug]	ログレベル 40 として出力します		
		（プレフィックスなし）	プレフィックス[Information]と同じ		
その他の機能	スクリプトで例外がスローされた場合、またはスクリプトで予期しない例外が発生した場合、部品が異常終了し、タスクログに例外が表示されます。				

	スクリプトが適切に終了し、スクリプトから返された値またはスクリプトの部品出力プロパティ（out0～out9）に指定された値に null または未定義の値が含まれる場合、それらの値は null または未定義として格納されます。

サンプルコード（scriptBody）

次の例では、Configuration Manager REST API（HTTP）のサンプルコードと、部品プロパティの JavaScript 本文に指定できる JavaScript のサンプルコードを示します。

プール情報を取得する

```
[Configuration Manager REST API（HTTP）の場合]
HTTP Method: GET
Request URL: ../ConfigurationManager/v1/objects/storages/
<storageDeviceID>/pools?poolType=<poolType>
Request Body: None

[JavaScript Plug-in for Configuration Manager REST API の場合]
var client = new api.ObjectsApi();
var arg = new
argDef.ObjectsApi.v1.objects.storages.storageDeviceID.pools.get();
arg.setStorageDeviceID(<storageDeviceID>);
arg.setDetailInfoType(<detailInfoType>);
arg.setPoolType(<poolType>);
arg.setResponseMaxWait(<responseMaxWait>);
arg.setResponseJobStatus(<responseJobStatus>);

client.v1.objects.storages.storageDeviceID.pools.get(arg);
```

プール情報を登録する

```
[Configuration Manager REST API（HTTP）の場合]
HTTP Method: POST
Request URL: ../ConfigurationManager/v1/objects/storages/
<storageDeviceID>/pools
Request Body: { "poolId": 76, "poolName": "pool1", "ldevIds": [405],
"poolType": "HDP" }

[JavaScript Plug-in for Configuration Manager REST API の場合]
var reqBody = {
  poolId:76,
  poolName:"pool1",
  ldevIds:[405],
  poolType:"HDP"
};

var client = new api.ObjectsApi();
var arg = new
argDef.ObjectsApi.v1.objects.storages.storageDeviceID.pools.post();
arg.setStorageDeviceID(<storageDeviceID>);
arg.setRequestBody(reqBody);
arg.setResponseMaxWait(<responseMaxWait>);
arg.setResponseJobStatus(<responseJobStatus>);

client.v1.objects.storages.storageDeviceID.pools.post(arg);
```

サンプルコード（scriptBody／importedScript）

次の例では、部品プロパティ「scriptBody」「importedScript」に指定できる Configuration Manager REST API のサンプルコードを示します。

```
-----------------------------------------
(scriptBody)
-----------------------------------------
function fn(serviceProperties, pluginProperties, arg0, arg1, arg2, arg3,
```

```
arg4, arg5, arg6, arg7, arg8, arg9) {

  hoge(CNST);

}
-----------------------------------------
(importedScript)
-----------------------------------------
var CNST = "hoge";

function hoge(a){
  print(a + " from common js!");
}
-----------------------------------------
```

B.18 Python 実行部品

Python 実行部品は、Ops Center Automator のホスト上で、Python スクリプトを実行するコンポーネントです。Python インタプリタは、Ops Center Automator によりインストールされないため、Python スクリプトを使用するには、使用する前に Ops Center Automator のホスト上に Python インタプリタをインストールする必要があります。この実行部品をクラスタ環境で使用するには、Python インタプリタを実行系および待機系システムの両方にインストールする必要があります。Python 実行部品は、Python の仮想環境をサポートしていません。

サポートされる Python のバージョン

バージョン 3.x シリーズ

終了コード

Python 実行部品は、以下の終了コードを生成します。

終了コード	説明
0	正常に終了しました。
1	Python インタプリタの実行に失敗しました。
2	Python スクリプトの実行に失敗しました。
3	Python スクリプトがタイムアウトしました。
80	タスクの実行が停止しました。
127	そのほかのエラーが発生しました。

プロパティリスト

Python 実行部品では、次のプロパティを使用できます。

プロパティキー	プロパティ名	説明	デフォルト値	入出力タイプ	必須
pythonInterpreterPath	Python インタプリタパス	スクリプトを実行する Python インタプリタへのパスを指定します。	python	入力	必須
scriptBody	スクリプト本体	Python コード文字列を指定します。	--	入力	必須
importedScript	インポートスクリプト	同じサービステンプレートに配置されたほかの Python 実行部品と共通で使用されるメソッドと定数	--	入力	任意

プロパティキー	プロパティ名	説明	デフォルト値	入出力タイプ	必須
		（Python のコード文字列）を指定します。 importedScript のスクリプトでは、次の項目を使用できます。 ・ 関数：log()関数。scriptBody で使用できる log()関数と同じです。 ・ 環境変数：scriptBody で使用できる環境変数と同じです。			
webServiceConnectionCategory	Web サービス接続先カテゴリ	Web サービス接続カテゴリを指定します。	--	入力	任意
webServiceConnectionName	Web サービス接続先名	Web サービス接続名を指定します。	--	入力	任意
timeout	タイムアウト	指定したスクリプトに対するタイムアウト時間を（秒単位で）指定します。	300	入力	任意
in*N*	スクリプト入力(*N*)	スクリプトに渡す引数を指定します。	--	入力	任意
standardOutput	標準出力	指定したスクリプトの標準出力を文字列として出力します。	--	出力	任意
standardErrorOutput	標準エラー出力	指定したスクリプトの標準エラー出力を文字列として出力します。	--	出力	任意
out*N*	スクリプト出力(*N*)	指定したスクリプトの"out*N*"関数の引数に、指定した値を出力します。	--	出力	任意
注※　*N*：0 から 9 の範囲の整数					

プロパティリストで部品の入力／出力プロパティを指定します。入力プロパティには、サービスプロパティの値、予約プロパティの値、およびリテラル文字の組み合わせを使用できます。

スクリプトで使用できる変数および関数

次の変数および関数をスクリプトで使用できます。

カテゴリ	名前	説明		
変数	in*N*	入力プロパティ（in*N*）に指定した値が設定されます。 値が配列型や JSON 形式で指定されていた場合も文字列型として解釈されます。		
関数	out*N*（文字列値）	関数の引数に渡された値が部品の出力プロパティ（out*N*）に出力されます。		
関数	log（文字列値）	任意の文字列をタスクログに出力できます。この場合、特定のプレフィックスを文字列の先頭に追加することで、ログレベルを選択できます。		
		プレフィックス	[Severe]	ログレベル 0 として出力します
			[Information]	ログレベル 10 として出力します

カテゴリ	名前	説明		
			[Fine]	ログレベル 20 として出力します
			[Finer]	ログレベル 30 として出力します
			[Debug]	ログレベル 40 として出力します
			（プレフィックスなし）	プレフィックス[Information]と同じ
注※　*N*：0 から 9 の範囲の整数				

デフォルトで定義された以下の import 文をスクリプトから削除すると、前述の変数および関数を使用できません。

```
from dnaplugin import in0, in1, in2, in3, in4, in5, in6, in7, in8, in9
from dnaplugin import out0, out1, out2, out3, out4, out5, out6, out7,
out8, out9
from dnaplugin import log
```

スクリプトから参照できる環境変数

スクリプト実行時に環境変数に値を設定します。os.environ.get(キー*名*)または os.environ[キー*名*]の形式で、以下の環境変数の値を取得できます。

環境変数	説明	フォーマット
PLUGIN_PROPERTIES	Python 実行部品のプロパティ	JSON 形式 {*プロパティ名*:*値*, ...}
SERVICE_TEMPLATE_ID	Python 実行部品が属するサービステンプレートの ID	数値
SERVICE_ID	Python 実行部品を実行するサービスの ID	数値
SERVICE_TEMPLATE	Python 実行部品が属するサービステンプレートの情報	JSON 形式 {*サービステンプレート属性*:*値*, ...}
SERVICE	Python 実行部品を実行するサービスの情報	JSON 形式 {*サービス属性*:*値*, ...}
WEB_SERVICE_CONNECTIONS	Web サービス接続の設定情報。入力プロパティに指定された「Web サービス接続先カテゴリ」および「Web サービス接続先名」に対応した情報が格納されます。格納される情報は入力プロパティの指定条件によって異なります（下表参照）。	JSON 形式 [{ *Web サービス接続属性*: *値*, ... }, ...]

入力プロパティ		参照情報
Web サービス接続カテゴリ	Web サービス接続名	
指定あり	指定あり	指定したカテゴリと名前に一致する Web サービス接続情報

入力プロパティ		参照情報
Web サービス接続カテゴリ	Web サービス接続名	
指定あり	指定なし	指定したカテゴリに一致する Web サービス接続情報
指定なし	指定あり	なし
指定なし	指定なし	なし

サンプルコード（scriptBody／importedScript）

次の例では、部品プロパティ「scriptBody」「importedScript」に指定できる Python 実行部品のサンプルコードを示します。

```
-----------------------------------------
(scriptBody)
-----------------------------------------
# -*- coding: utf-8 -*-
import os
from dnaplugin import in0, in1, in2, in3, in4, in5, in6, in7, in8, in9
from dnaplugin import out0, out1, out2, out3, out4, out5, out6, out7,
out8, out9
from dnaplugin import log
from dnaplugin_imported_script import *

hoge(CNST)
-----------------------------------------
(importedScript)
-----------------------------------------
from dnaplugin import log

def hoge(a):
  log(a + ' from common py!')

CNST = 'hoge'
-----------------------------------------
```

B.19 Web クライアント部品

Web クライアント部品は、HTTP リクエストとレスポンスメッセージを送信および受信します。リクエストされると、プロキシ経由でサーバへアクセスし、サーバとプロキシの認証を行います。

機能

この部品は、HTTP リクエストとレスポンスメッセージを送信および受信し、次の機能で構成されます。

- HTTP/HTTPS 1.1 をサポートします。
- 入力プロパティに基づいて HTTP リクエストメッセージを生成し、出力プロパティとして HTTP レスポンスメッセージを受信します。

HTTP リクエストメッセージと、それらに対応する入力プロパティとの関係を、次の表に示します。

親要素		入力プロパティ
リクエストライン	Method	requestMethod
	URI	requestUrl

親要素		入力プロパティ
	HTTP/version	-
ヘッダー		requestHeaders
本文		requestBody

HTTP レスポンスメッセージと、それらに対応する出力プロパティとの関係を、次の表に示します。

親要素		出力プロパティ
ステータスライン	HTTP/version	-
	ステータスコード	responseStatusCode
	ステータスメッセージ	responseStatusMessage
ヘッダー		responseHeaders
本文		responseBody

実行の前提条件

- HTTPS 通信を使用するには、通信先サーバの証明書、またはサーバ証明書を認証する root 証明書を、クライアント（Ops Center Automator インストールサーバ）の JRE Trust Store へインポートします。

終了コード

Web クライアント部品では、次の終了コードが生成されます。

終了コード	説明
0	正常に終了しました。
1	HTTP レスポンスメッセージに、Success 以外のステータスコードが返されました。
70	相手ホストへの接続に失敗しました。
77	ホスト名を相手ホストに解決できませんでした。
80	タスクの実行が停止しました。
86	不正なプロパティ値が指定されました。
90	接続の確立後に、データ転送が失敗しました。
91	HTTP レスポンスメッセージのサイズが、システムの上限値を超えています。
127	指定されていない種類のエラーが発生しました。

プロパティリスト

Web クライアント部品では、次のプロパティを使用できます。詳細は「表 49 Web サービス接続の値と Web クライアント部品の入力値との関係」を参照してください。

プロパティキー	プロパティ名	説明	入出力タイプ
webServiceConnectionCategory	Web サービス接続先定義のカテゴリ	Web サービス接続の情報を参照してその情報を Web クライアント部品の入力値として	入力

プロパティキー	プロパティ名	説明	入出力タイプ
		使用する場合、Web サービス接続のカテゴリを指定します。	
webServiceConnectionName	Web サービス接続先定義の接続先名	Web サービス接続の情報を参照してその情報を Web クライアント部品の入力値として使用する場合、Web サービス接続の名前を指定します。	入力
requestMethod※	メソッド	HTTP メソッドを次のように指定します。 ・ GET ・ POST ・ PUT ・ DELETE デフォルト値は GET です。	入力
requestUrl※	リクエスト URL	HTTP リクエストの送信先リソースの URL と、クエリパラメタを指定します。入力プロパティ値として指定された URL のエンコード値は、そのまま使用されます。詳細は「表 49 Web サービス接続の値と Web クライアント部品の入力値との関係」を参照してください。	入力
requestHeaders	リクエストヘッダ	HTTP リクエストヘッダーを raw 形式で指定します。リクエストボディの形式と文字セットを指定するには、Content-Type ヘッダーと charset パラメーターを使用します。	入力
requestBody	リクエストボディ	HTTP リクエストの本文を raw 形式で指定します。Content-Type ヘッダーに指定した形式を使用します。	入力
webAuth※	サーバ認証スキーム	サーバ認証の種別を指定します。 ・ none -- 認証なし ・ basic -- Basic 認証 ・ digest -- Digest 認証 ・ negotiate -- Negotiate 認証 `requestHeaders` の認証ヘッダーには "none"を指定します。	入力
webUsername	サーバ認証ユーザー名	サーバ認証に使用するユーザー名を、最大 256 文字で指定します。webAuth プロパティキーが none に設定されているときは無効です。詳細は「表 49 Web サービス接続の値と Web クライアント部品の入力値との関係」を参照してください。	入力
webPassword	サーバ認証パスワード	サーバ認証に使用するパスワードを、最大 256 文字で指定します。webAuth プロパティキーが none に設定されているときは無効です。詳細は「表 49 Web サービス接続の値と Web クライアント部品の入力値との関係」を参照してください。	入力
useProxy※	プロキシの使用	プロキシ接続が必要な場合は true に設定します。詳細は「表 49 Web サービス接続の値	入力

プロパティキー	プロパティ名	説明	入出力タイプ
		と Web クライアント部品の入力値との関係」を参照してください。 デフォルト値は false です。	
proxyHostname	プロキシホスト名	プロキシのホスト名または IP アドレスを、最大 256 文字で指定します。useProxy プロパティキーが false に設定されているときは無効です。詳細は「表 49 Web サービス接続の値と Web クライアント部品の入力値との関係」を参照してください。	入力
proxyPort	プロキシポート番号	プロキシのポート番号を、1～65535 の範囲で指定します。useProxy プロパティキーが false に設定されているときは無効です。詳細は「表 49 Web サービス接続の値と Web クライアント部品の入力値との関係」を参照してください。 デフォルト値は 8080 です。	入力
proxyAuth※	プロキシサーバ認証スキーム	プロキシ認証の種別を指定します。次の認証機能がサポートされています。 ・ none -- 認証なし ・ basic -- Basic 認証 ・ digest -- Digest 認証 requestHeaders の認証ヘッダーには"none"を指定します。詳細は「表 49 Web サービス接続の値と Web クライアント部品の入力値との関係」を参照してください。 デフォルト値は none です。	入力
proxyUsername	プロキシユーザー名	プロキシ認証に使用するユーザー名を、最大 256 文字で指定します。useProxy プロパティキーが false に設定されている場合、または proxyAuth プロパティキーが none に設定されている場合は無効です。詳細は「表 49 Web サービス接続の値と Web クライアント部品の入力値との関係」を参照してください。	入力
proxyPassword	プロキシパスワード	プロキシ認証に使用するパスワードを、最大 256 文字で指定します。useProxy プロパティキーが false に設定されている場合、または proxyAuth プロパティキーが none に設定されている場合は無効です。詳細は「表 49 Web サービス接続の値と Web クライアント部品の入力値との関係」を参照してください。	入力
responseStatusCode	ステータスコード	HTTP レスポンスメッセージのステータスコードを出力します。"3xx Redirect"が返されたとき、自動トラッキングが有効になります。	出力
responseStatusMessage	ステータスメッセージ	HTTP レスポンスメッセージのステータスメッセージを出力します。	出力
responseHeaders	レスポンスヘッダ	HTTP レスポンスメッセージのヘッダー情報を出力します。	出力

プロパティキー	プロパティ名	説明	入出力タイプ
responseBody	レスポンスボディ	HTTP レスポンスメッセージの本文情報を出力します。responseBody のサイズが 30MB を超えないようにしてください。responseBody のサイズが 30MB を超えると、その値は切り捨てられます。	出力

注※　必須プロパティです。

メモ　Basic 認証を使用するとき、認証なしのリクエストではレスポンスコード 401 と、認証がリクエストされることを示す「WWW-Authenticate:Basic」ヘッダーが返されます。「WWW-Authenticate:Basic」が返されないとき、Base64 変換された「Username:Password」を含む認証トークンがリクエストヘッダーに追加されます。Digest 認証では、nonce 値などの値がサーバから返される必要があるため、プリエンプティブ手法による認証はサポートされません。

Web サービス接続の設定値を、Web クライアント部品の入力プロパティの値として参照できます。この場合、対応する入力プロパティは Web サービス接続の設定値で上書きされます。

Web サービス接続の設定値を参照するかどうかは、webServiceConnectionCategory と webServiceConnectionName の入力値で決まります。

webServiceConnectionCategory の入力値	webServiceConnectionName の入力値	Web クライアント部品の動作
あり	あり	Web サービス接続の値を参照します。一致する Web サービス接続が見つからない場合、実行時エラーになります。
あり	なし	エラー（Web サービス接続の参照を試みましたが、一致するものが見つかりませんでした。）
なし	あり	エラー（Web サービス接続の参照を試みましたが、一致するものが見つかりませんでした。）
なし	なし	Web サービス接続の値を参照しません。

Web サービス接続の設定値を参照する場合、URL（プロパティ名：requestUrl）は、実行時に動的に組み立てられます。プロパティの入力値として、ホスト名のあとの「/」の後ろの部分を指定する必要があります。「/」の前の部分は、Web サービス接続の設定値から組み立てられます。requestUrl に指定した値が「/」で始まっていない文字列の場合、実行時エラーが生成されます。

例："http://host:port/Folder/"の場合、「/Folder/」を入力値として指定します。Web サービス接続の値と Web クライアント部品の入力値との関係を、次の表に示します。

表 49 Web サービス接続の値と Web クライアント部品の入力値との関係

Web サービス接続の項目	Web クライアント部品の入力プロパティ	備考
IP アドレス/ホスト名	requestUrl の一部	"http://host:port/Folder/"の host の部分

Web サービス接続の項目	Web クライアント部品の入力プロパティ	備考
プロトコル	requestUrl の一部	"http://host:port/Folder/"の http の部分
ポート番号	requestUrl の一部	"http://host:port/Folder/"の port の部分
ユーザー ID	webUsername	入力値が設定されている場合、上書きされることを示すメッセージを、タスクログに出力します。
パスワード	webPassword	入力値が設定されている場合、上書きされることを示すメッセージを、タスクログに出力します。
プロキシ設定	useProxy	−
プロキシサーバの IP アドレス/ホスト名	proxyHostname	入力値が設定されている場合、上書きされることを示すメッセージを、タスクログに出力します。
プロキシサーバのポート番号	proxyPort	入力値が設定されている場合、上書きされることを示すメッセージを、タスクログに出力します。
プロキシサーバの認証方式	proxyAuth	−
プロキシサーバのユーザー ID	proxyUsername	入力値が設定されている場合、上書きされることを示すメッセージを、タスクログに出力します。
プロキシサーバのパスワード	proxyPassword	入力値が設定されている場合、上書きされることを示すメッセージを、タスクログに出力します。

サポートされるヘッダー

ヘッダーは raw 形式で転送され、受信されます。次のデフォルトヘッダーがサポートされています。

ヘッダー	値
Accept	application/json
Accept-Language	en
Content-Type（メソッドとして POST または PUT が指定された場合のみ）	application/json
Cache-Control	no-cache
Pragma	no-cache
User-Agent	client-software-name, version
Host	destination-host-name, port-number
Connection	keep-alive

次のヘッダーは特別な動作を行います。

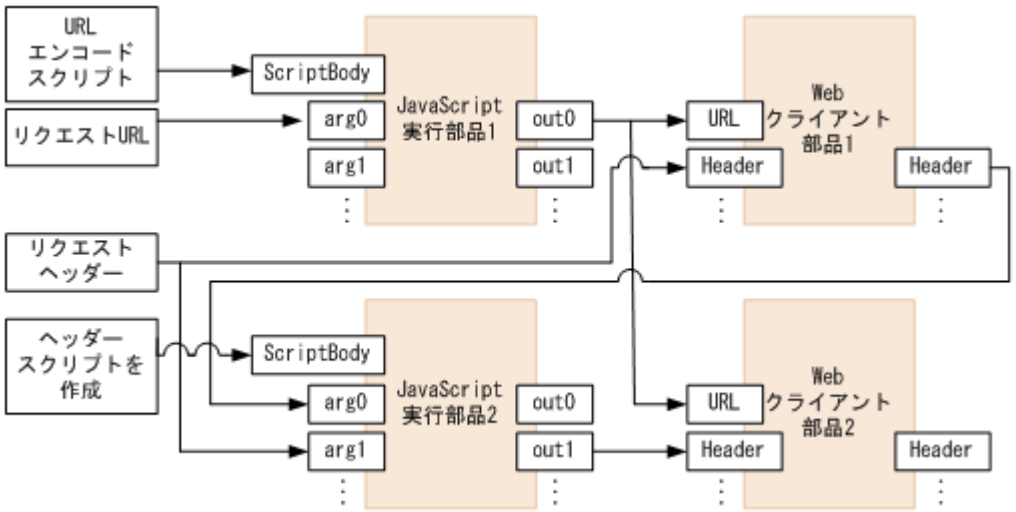
ヘッダー	動作
Content-Type ヘッダーの Charset パラメーター	リクエスト:Content-Type ヘッダーの Charset パラメーターに指定されている値を使用して、本文の文字セットが変換されます。値が指定されていない場合、UTF-8 に変換されます。
	レスポンス:Content-Type ヘッダーの Charset パラメーターに指定されている値に従って、本文の文字セットが解釈されます。charset パラメーターが返されない場合、UTF-8 として解釈されます。
Content-Encoding	Content-Encoding ヘッダーが返された場合、本文が展開されます。次のエンコード形式がサポートされています。 • gzip • deflate

接続タイムアウト値の設定

通信先へ接続するとき発生の可能性がある HTTP/HTTPS 通信の問題を処理する場合、接続タイムアウトキーの値（plugin.http.connect.timeout）を構成し、接続に問題が発生したことを明確にしてください。接続タイムアウト値は、*Automation-software-installation-folder*¥conf の下のプロパティファイルで、キー名（config_user.properties）により指定します。

JavaScript 実行部品との連携

Web クライアント部品は入力および出力のプロパティ値を書き替えないため、値の書き替えが必要な場合は JavaScript 実行部品との連携が効果的です。JavaScript 実行部品との連携により、URL エンコードと SSO 認証トークンをサーバレスポンスから抽出する例を、以下に示します。



次の表は、JavaScript 実行部品とのフロー連携を示したものです。

部品	フロー	説明
JavaScript 実行部品 1	入力	ユーザーにより入力された URL。URL エンコーディングを実行するスクリプト。

部品	フロー	説明
	実行	URL エンコーディングを実行します。
	出力	URL エンコードされた URL。
Web クライアント部品 1	入力	URL エンコードされた URL。 ユーザーにより入力された他の情報（ヘッダーなど）。
	実行	HTTP リクエストを生成し送信します。 HTTP レスポンスを受信し解析します。
	出力	HTTP レスポンスの親要素（ヘッダーなど）を出力します。
JavaScript 実行部品 2	入力	URL エンコードされた URL。 再構成されたヘッダー。 ユーザーにより入力された他の情報。
	実行	HTTP リクエストを生成し送信します。 HTTP レスポンスを受信し解析します。
	出力	HTTP レスポンスの親要素を出力します。

Negotiate 認証を使用するための前提条件

Negotiate 認証は Kerberos v5 認証を使用するため、次の構成ファイルが必要です。

ファイル名	説明	ユーザーによる編集
Kerberos 構成ファイル (krb5.conf)	ユーザー環境での Kerberos 構成	必須
ログイン構成ファイル (login.conf)	使用される認証テクノロジーを指定します。	オプション

以下に示すように、Kerberos 構成ファイルは*Automation-software-installation-folder*¥conf¥plugin に置かれます。ユーザー環境での必要に応じて、コードの編集、特に斜体の文字を編集します。

```
[libdefaults] // Default value for Kerberos authentication
 default_realm = EXAMPLE.COM // Default realm for Kerberos authentication
 udp_preference_limit = 1

[realms] // KDC setting for each Kerberos realm (you can define settings
for multiple realms)
 EXAMPLE.COM = {
   kdc = kdc.example.com // KDC host name
 }

[domain_realm] // Maps the Active Directory domain to the Kerberos realm
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```

メモ　レルムでは大文字と小文字が区別されます。慣習的に大文字が使用されるため、レルムを指定するには大文字を使用します（小文字は使用できません）。

メモ　Kerberos 認証はデフォルトで UDP を使用しますが、大きなメッセージには TCP を使用します。UDP を使用する TGT リクエストは Active Directory とのリンクに失敗するため、udp_preference_limit=1 を設定して TCP を使用します。

ログイン構成ファイル

以下のログイン構成ファイルが *Automation-software-installation-folder*¥conf¥plugin に置かれます。

```
com.sun.security.jgss.krb5.initiate {
  com.sun.security.auth.module.Krb5LoginModule required
useTicketCache=true doNotPrompt=false refreshKrb5Config=true; //
Specifies the authentication
technology and options to be used
};
```

Common-Component-installation-folder¥uCPSB11¥CC¥web¥containers¥AutomationWebService¥usrconf¥usrconf.cfg に次の内容を入力し、Kerberos 構成ファイルとログイン構成ファイルが参照されるようにします。

```
add.jvm.arg=-Djava.security.krb5.conf=Automation/conf/plugin/krb5.conf
add.jvm.arg=-Djava.security.auth.login.config=Automation/conf/plugin/
login.conf add.jvm.arg=-Djavax.security.auth.useSubjectCredsOnly=false
```

索引

に

は

ひ

ふ

め

も

り

れ