Informatica® Cloud Application Integration
October 2023

invoke

Informatica Cloud Application Integration invoke
October 2023

発行日: 2023-11-28

目次

序文

「*呼び出し*」を使用して、アプリケーションの統合アセットをデプロイする方法を確認します。

第 1 章

プロセスのランタイムタスク

プロセスを作成した後、以下のランタイムタスクを実行できます。

プロセスのパブリッシュ

プロセスを呼び出したり実行するには、プロセスをパブリッシュする必要があります。プロセスをパブリッシュすると、アプリケーション統合ではサービスエンドポイント URL、Swagger ファイル、OpenAPI 3.0 ファイル、WSDL ファイルがプロセスに対して生成されます。プロセスを呼び出すには、サービスエンドポイント URL を使用します。プロセスの API 定義を表示するには、Swagger ファイル、OpenAPI 3.0 ファイル、WSDL ファイルを使用します。

プロセスの呼び出し

プロセスを呼び出すには、プロセスをクラウドサーバー、特定の Secure Agent、または Secure Agent グループにデプロイします。さまざまな HTTP 動詞を使ってプロセスを呼び出すことができます。

プロセス入力を作成し、プロセス入力を使用してプロセスを実行します。

プロセスをパブリッシュし、プロセス入力を作成した後、プロセスをプロセス入力を使用して実行し、テストすることができます。プロセスを実行した後、プロセスインスタンス実行の成功と失敗の詳細を表示できます。

プロセスのパブリッシュ

プロセスを呼び出したり実行するには、プロセスをパブリッシュする必要があります。

複数のプロセスをまとめてパブリッシュすることもできます。詳細については、第 3 章,「アプリケーション統合アセットの一括パブリッシュ」 (ページ 29)を参照してください。

プロセスをパブリッシュすると、アプリケーション統合によって API がアクティブ化されます。また、プロセスのサービス URL、Swagger ファイル、OpenAPI 3.0 ファイル、および WSDL ファイルが生成されます。プロセスを呼び出すには、サービスエンドポイント URL を使用します。プロセスの API 定義を表示するには、Swagger ファイル、OpenAPI 3.0 ファイル、WSDL ファイルを使用します。

パブリッシュされたプロセスを編集する場合、変更を反映させるためにプロセスを再度パブリッシュする必要があります。そうしなかった場合、**[プロパティの詳細]** ダイアログボックスと **[参照]** ページで、そのプロセ

スのステータスが**［古くなっている］**に変わります。**［古くなっている］**ステータスは、プロセスにパブリッシュされていない変更が含まれているということです。

1. **［参照］**ページで、パブリッシュするプロセスに移動し、**［パブリッシュ］**をクリックします。

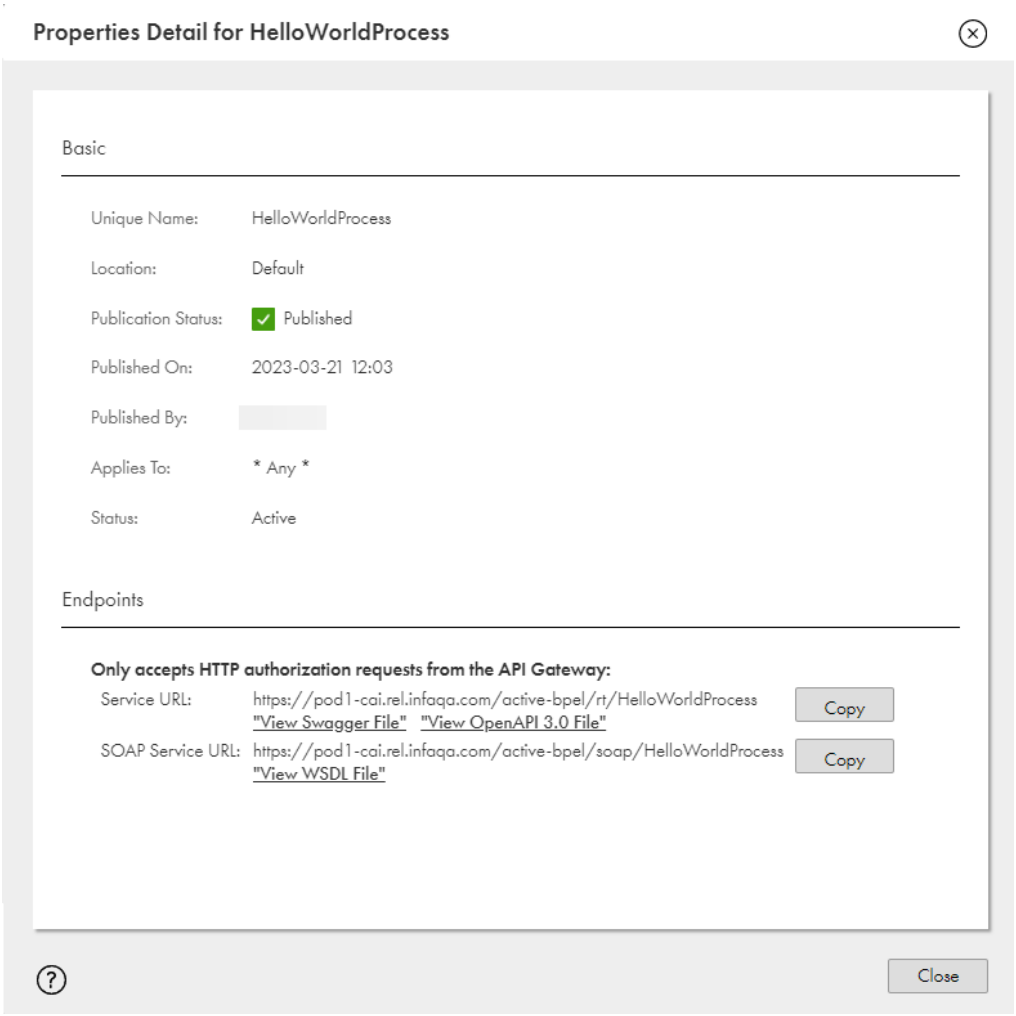
   アプリケーション統合でプロセスがパブリッシュされ、API がアクティブ化されます。また、サービス URL、Swagger ファイル、OpenAPI 3.0 ファイル、および WSDL ファイルが生成されます。

   注: 許可されたグループまたはユーザーを指定しない場合、またはプロセスに対する匿名アクセスを許可する場合、アプリケーション統合ではサービス URL、Swagger ファイル、OpenAPI 3.0 ファイル、WSDL ファイルは生成されません。

2. **［アクション］**メニューで、**［プロパティの詳細］**をクリックして、API ステータス、生成されたサービス URL、Swagger ファイル、OpenAPI 3.0 ファイル、WSDL ファイルを表示します。

   ［プロパティの詳細］ダイアログボックスが表示されます。

   以下の図は、**［プロパティの詳細］**ダイアログボックスを示しています。



   ステータスは、API がアクティブであることを示しています。

   注: アプリケーションの統合から初めてプロセスをパブリッシュすると、API がアクティブ化され、アプリケーション統合コンソールの **［API］**ページに表示されます。後ほど **［API］**ページから API をアクティブ化または非アクティブ化すると、アプリケーションの統合によって **［プロパティの詳細］**ダイアログボックスの API ステータスも更新されます。API ステータスが非アクティブであるプロセスを再パブリッシュしても、API はアクティブ化されません。ユーザーが API を使用できるようにするには、**［API］**ページから API をアクティブ化する必要があります。API のアクティブ化と非アクティブ化の詳細については、Monitor のヘルプの「*API*」を参照してください。

   また、サービス URL および SOAP サービスの URL を表示し、それらを使用してプロセスを呼び出すこともできます。**［エンドポイント］**セクションは、プロセスが API ゲートウェイからの HTTP 認証要求のみを承認するように設定されているかどうかを示します。

   関連付けられている Swagger ファイル、OpenAPI 3.0 ファイル、および WSDL ファイルを表示するには、**［Swagger ファイルの表示］**リンク、**［OpenAPI 3.0 ファイルの表示］**リンク、および **［WSDL ファイルの表示］**リンクをクリックします。

プロセスのパブリッシュ解除

プロセス名または API 名を編集したり、パブリッシュされたプロセスを無効にしたりするには、プロセスをパブリッシュ解除する必要があります。パブリッシュされていないプロセスをサービスエンドポイントの URL から呼び出すことはできません。したがって、プロセスをパブリッシュ解除した後で、それに応じて関連する API クライアントを更新する必要があります。

1. **［参照］**ページで、パブリッシュ解除するプロセスに移動します。
2. **［アクション］**メニューで、**［パブリッシュ解除］**をクリックします。

   アプリケーション統合でプロセスがパブリッシュ解除され、アプリケーション統合コンソールの**［API］**ページから API が削除されます。また、サービス URL、Swagger ファイル、OpenAPI 3.0 ファイル、および WSDL ファイルが無効になります。

必要な変更を加えた後に、プロセスを再度パブリッシュして API をアクティブ化し、サービス URL、Swagger ファイル、OpenAPI 3.0 ファイル、および WSDL ファイルを生成します。プロセスをパブリッシュすると、プロセスが再び有効になります。

クラウドサーバーにデプロイしたプロセスの呼び出し

以降のトピックでは、クラウドサーバーにデプロイした Process Designer プロセスおよび Process Developer プロセスを呼び出す方法について説明します。

Web サービスと SOAP エンドポイント

Process Designer は、自動生成された WSDL インタフェースを通じて Web サービスを公開します。ここにアクセスして、SOAP エンドポイントとして使用できます。使用する各 Web サービスには、アプリケーションサーバーでホストされているメソッドのセットが含まれており、これらは操作とも呼ばれます。これらの操作を、クラウド上で、またはネットワークを経由してリモートから起動できます。操作が起動したら、SOAP メッセージを作成し、それを通常は HTTP/HTTPS を経由して Web サービスに送信します。SOAP は、一般的な XML ベースのメッセージングプロトコルで、情報を交換するために使用します。

Web サービスには、操作自体とそれらの起動方法を記述した XML である、WSDL（Web Services Definition Language）ドキュメントが含まれます。使用するすべての Web サービスには、サービス名とエンドポイント URL（操作を起動するために SOAP メッセージを送信する場所）が必要です。

使用可能な操作は SOAP インタフェースによってパブリッシュされるため、Web サービスの呼び出し元では内部的な詳細を把握しておく必要はありません。それぞれの呼び出しに対して、Web サービスは応答メッセージを返します。応答には、要求された情報またはフォールト情報（エラーの場合）が含まれます。

次の簡単な SOAP メッセージは、「追加」メソッドを呼び出して応答を返します。

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
    xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xmlns:s='http://www.w3.org/2001/XMLSchema'>
    <SOAP-ENV:Body>
        <Add xmlns="http://www. math.org">
            <arg1 xsi:type="s:decimal">100</arg1>
            <arg2 xsi:type="s:decimal">25</arg2>
        </Add>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

この場合、応答メッセージは次のようになります。

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
    xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xmlns:s='http://www.w3.org/2001/XMLSchema'>
    <SOAP-ENV:Body>
        <AddResponse xmlns="http://www.math.org">
            <AddResult>125</AddResult>
        </AddResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Web サービスとして公開されるプロセスの定義

Process Designer を使用してプロセスを定義するときに、"xsd:any"のペイロードスタイルを処理する SOAP クライアントは SOAP エンドポイントを使用できます。これは、クライアントが入力と出力を処理し、フォールトを扱えることを意味します。

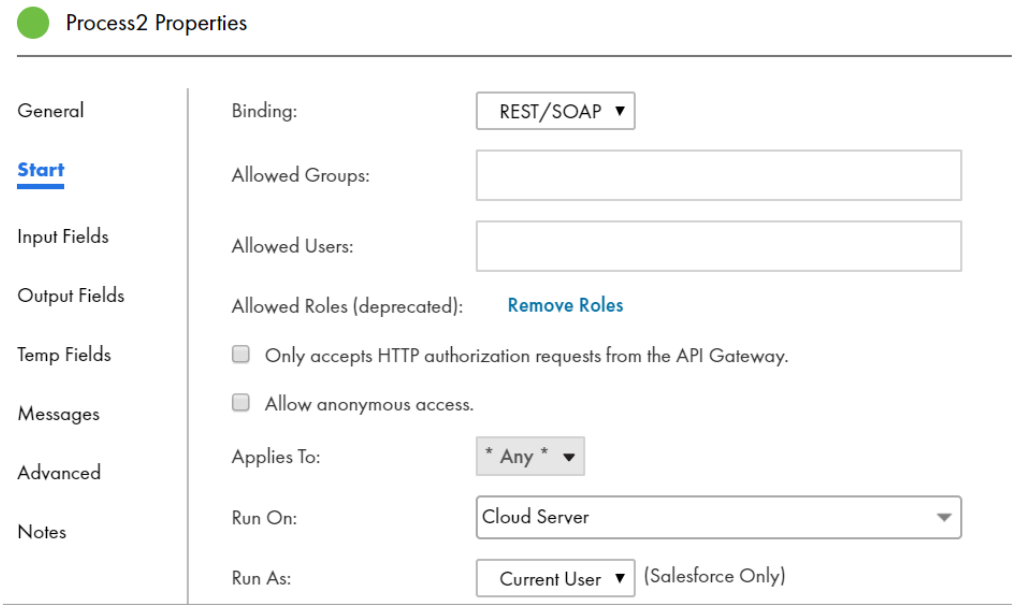
Process Designer は、ドキュメント/リテラルのメッセージスタイルと、ほとんどの Web サービスが使用するエンコーディングオプションを備えた SOAP 1.1 標準をサポートします。

WSDL インタフェースでは、次のことができます。

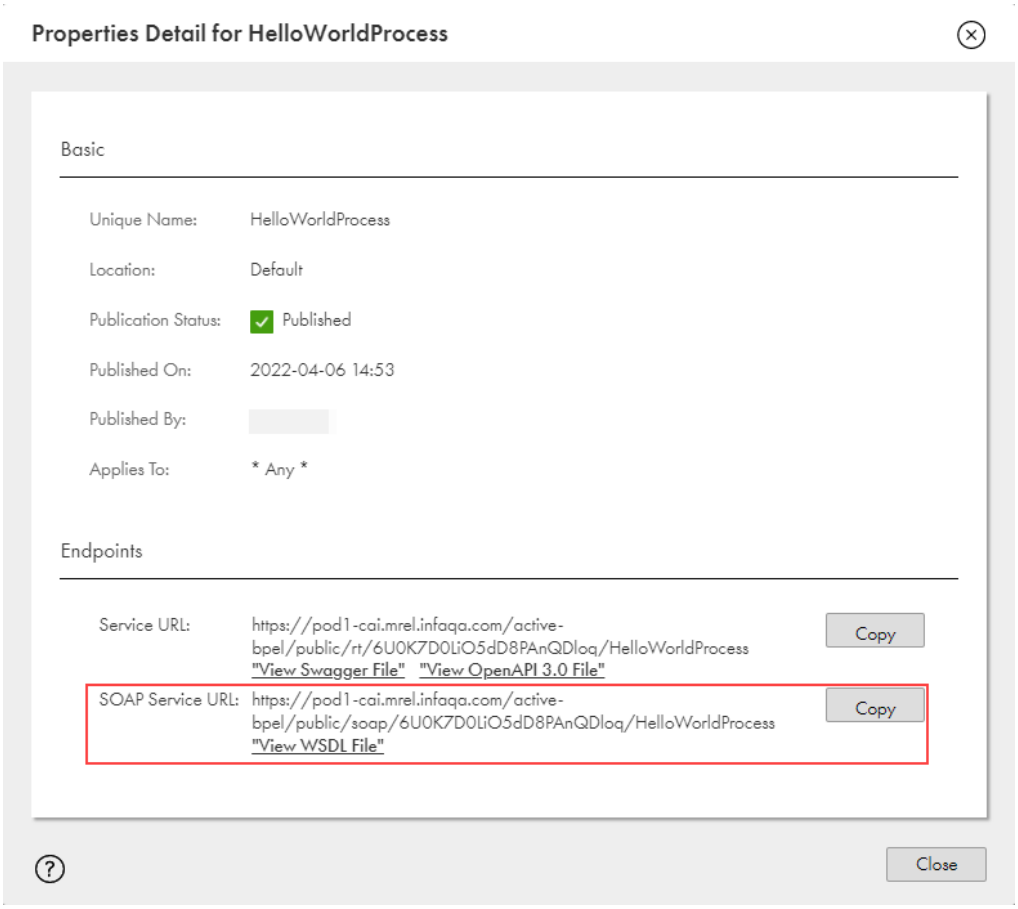
- プロセスの入力フィールドと出力フィールドまたはプロセスオブジェクトに基づいた、インラインスキーマを使用した、WSDL 入力タイプと出力タイプの定義。名前空間は自動的に生成されます。
- SOAP サービスの URL のプロセス詳細への表示。
- プロセス詳細から WSDL 定義へのリンク。
- REST と整合した、Basic 認証の使用。
- スローステップを使用したフォールトの生成。これにより、1 つのフォールトを共有して説明を提供することが可能になります。
- インラインメッセージイベントと境界の条件を扱うための、WSDL での相関プロパティの生成。

SOAP エンドポイントを Process Designer で定義するには、次の手順を実行します。

1. プロセスのプロパティで、 **［バインディング］**の開始として**［REST/SOAP］**を選択します。
   次の図に、［REST/SOAP］として設定された**［バインディング］**プロパティを示します。



2. プロセスを定義してパブリッシュした後に、SOAP サービス URL が表示されます。
次の図に、SOAP サービス URL を示します。



3. **[WSDL ファイルの表示]** をクリックして、完全な WSDL を開き、このプロセス用に生成された SOAP メッセージを参照します。URL をコピーしてエンドポイントとして使用できます。

   注: プロセス名に日本語の文字が含まれている場合、Chrome ブラウザで WSDL ファイルを表示するとエラーが表示されることがあります。エラーが表示された場合、ページを右クリックし、**[ページのソースを表示]** を選択して、WSDL ファイルを表示します。

メッセージイベントにアクセスするための REST および Process Designer の SOAP エンドポイント

Process Designer が生成した REST および SOAP 1.1 のエンドポイントを使用して、メッセージイベントにアクセスできます。SOAP 1.2 エンドポイントが必要な場合は、Process Developer を使用してこれらのエンドポイントを生成します。Process Developer を使用して作成されたサービスのエンドポイントを取得する方法については、Process Developer のガイドに記載されているポリシーに基づいたバインディングに関する情報を参照してください。

次の表に、メッセージイベントを起動する際に必要なアクセスのタイプに基づいた、REST または SOAP 1.1 それぞれのエンドポイントのパターンを指定します。

メッセージイベントのプロセスの開始プロパティとメッセージプロパティで **[匿名アクセスを許可]** が選択解除されている場合、プロセスでは HTTP Basic アクセス認証が使用されます。メッセージイベントやサブプロ

セスの呼び出しで、プロセスが受け取るコールバックとは異なる方法で認証するようにプロセスを定義できます。

アクセスのタイプ	REST URL の形式	SOAP 1.1 URL の形式
HTTP Basic アクセス認証を使用するプロセス	https://{host}/active-bpel/rt/{ProcessName}	https://{host}/active-bpel/soap/{ProcessName}
OAuth2 認証を使用するプロセス	https://{host}/active-bpel/rt/{ProcessName}	https://{host}/active-bpel/soap/{ProcessName}
匿名アクセスを使用するプロセス	https://{host}/active-bpel/public/rt/{TenantName}/{ProcessName}	https://{host}/active-bpel/public/soap{TenantName}/{ProcessName}
匿名アクセスを使用するプロセスメッセージイベント	https://{host}/active-bpel/public/rt/{TenantName}/{ProcessName}/event/{MessageName}	https://{host}/active-bpel/public/soap/{TenantName}/{ProcessName}/event/{MessageName}
HTTP Basic アクセス認証を使用するプロセスメッセージイベント	https://{host}/active-bpel/rt/{ProcessName}/event/{MessageName}	https://{host}/active-bpel/soap/{ProcessName}/event/{MessageName}
匿名アクセスを使用するサブプロセス内のプロセスメッセージイベント	https://{host}/active-bpel/public/rt/{TenantName}/{SubProcessName}/event/{MessageName}	https://{host}/active-bpel/public/soap/{TenantName}/{ProcessName}/event/{MessageName} または https://{host}/active-bpel/public/soap/{TenantName}/{SubProcessName}/event/{MessageName}
HTTP Basic アクセス認証を使用するサブプロセス内のプロセスメッセージイベント	https://{host}/active-bpel/rt/{SubProcessName}/event/{MessageName}	https://{host}/active-bpel/soap/{ProcessName}/event/{MessageName} または https://{host}/active-bpel/soap/{SubProcessName}/event/{MessageName}

注: SOAP 要求で、イベントがプロセスまたはサブプロセスと同じアクセス方法（匿名または HTTP Basic アクセス認証）を使用する場合は、デフォルトのプロセスまたはサブプロセスのエンドポイントを使用できます。

Secure Agent にデプロイしたプロセスの呼び出し

Secure Agent 30.0 以降を使用している場合、HTTP エンドポイントおよび HTTPS エンドポイントを使用して、Secure Agent にデプロイしたプロセスを呼び出すことができます。

以降のセクションを参照し、REST エンドポイントおよび SOAP エンドポイントを作成するかそれらにアクセスして、Secure Agent にデプロイしたプロセスに HTTP または HTTPS を介してアクセスしてください。Informatica Process Designer（IPD）プロセスをデプロイする場合、匿名アクセス用のエンドポイント URL の構文は非匿名アクセス用のエンドポイント URL の構文とは異なります。

Process Designer プロセスの呼び出し

アクセスのタイプ	REST エンドポイント	SOAP 1.2 エンドポイント
HTTP を使用した IPD プロセスの非匿名アクセス。	http://[host][:port]/process-engine/rt/[serviceName] Swagger: http://[host][:port]/process-engine/rt/[serviceName]?swagger	http://[host][:port]/process-engine/soap/[serviceName] WSDL: http://[host][:port]/process-engine/soap/[serviceName]?wsdl
HTTP を使用した IPD プロセスの匿名アクセス。	http://[host][:port]/process-engine/public/rt/[serviceName] Swagger: http://[host][:port]/process-engine/public/rt/[serviceName]?swagger	http://[host][:port]/process-engine/public/soap/[serviceName] WSDL: http://[host][:port]/process-engine/public/soap/[serviceName]?wsdl
HTTPS を使用した IPD プロセスの非匿名アクセス。	https://[host][:port]/process-engine/rt/[serviceName] Swagger: https://[host][:port]/process-engine/rt/[serviceName]?swagger	https://[host][:port]/process-engine/soap/[serviceName] WSDL: https://[host][:port]/process-engine/soap/[serviceName]?wsdl
HTTPS を使用した IPD プロセスの匿名アクセス。	https://[host][:port]/process-engine/public/rt/[serviceName] Swagger: https://[host][:port]/process-engine/public/rt/[serviceName]?swagger	https://[host][:port]/process-engine/public/soap/[serviceName] WSDL: https://[host][:port]/process-engine/public/soap/[serviceName]?wsdl

Process Developer プロセスの呼び出し

次の手順に従って、呼び出すことのできる Process Developer SOAP および REST エンドポイント URL のリストを取得します。

1. 以下の **[Administrative Services（管理サービス）]** ページを開きます。
   - `https://localhost:7443/process-engine/`（HTTPS エンドポイント URL の場合）
   - `http://localhost:7080/process-engine/`（HTTP エンドポイント URL の場合）
2. **[Visit the Web Services Listing（Web サービスリスティングにアクセス）]** または **[Visit the REST Service（REST サービスにアクセス）]** をクリックします。
3. Informatica Cloud Services 管理者のユーザー資格情報を入力します。

SOAP または REST エンドポイント URL のリストが表示されます。SOAP または REST クライアントを使用して、これらのエンドポイント URL を呼び出します。

Secure Agent グループにデプロイしたプロセスの呼び出し

特定の Secure Agent または Secure Agent グループにデプロイしたプロセスを、SOAP/REST URL を介して呼び出すことができます。Secure Agent グループにプロセスをデプロイしている場合は、ロードバランシング URL を介してプロセスを呼び出すこともできます。

プロセスは次の方法で呼び出すことができます。

特定の Secure Agent **にデプロイしたプロセスを呼び出す**

プロセスを Secure Agent グループ内の特定の Secure Agent にデプロイできます。そうしたプロセスを呼び出すには、**[Design Home（設計ホーム）]** ページの SOAP/REST サービス URL を使用します。SOAP/REST サービス URL には、プロセスをデプロイした Secure Agent のプロパティが反映されています。

Secure Agent **グループにデプロイしたプロセスを呼び出す**

プロセスを Secure Agent グループにデプロイすることができます。そうしたプロセスを呼び出すには、**[Design Home（設計ホーム）]** ページの SOAP/REST サービス URL を使用します。SOAP/REST URL には、Secure Agent グループのマスタ Secure Agent のプロパティが反映されています。

Secure Agent グループにプロセスをデプロイしていてロードバランサがある場合、ロードバランサ URL を作成できます。**[Design Home（設計ホーム）]** ページの SOAP/REST サービス URL に加えて、ロードバランサ URL を介してプロセスを呼び出すことができます。

Secure Agent グループの詳細については、「*Administrator*」のヘルプの「*Secure Agent サービス*」を参照してください。

HTTP 動詞を使用したプロセスの呼び出し

次の HTTP 動詞を使用してプロセスを呼び出すことができます。

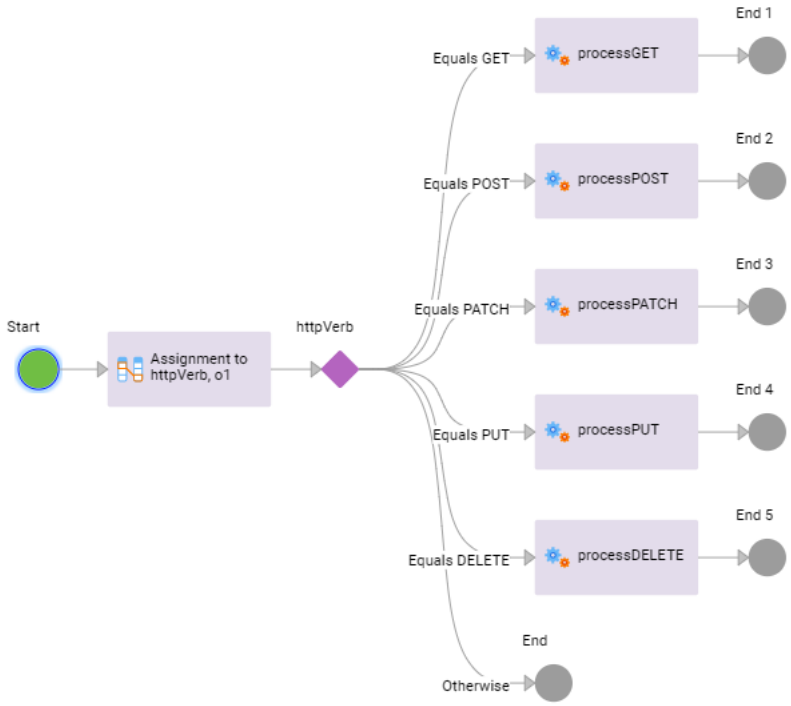
- GET
- POST
- PATCH
- PUT
- DELETE

さまざまな HTTP 動詞を使用し、要求で使用されている HTTP 動詞を基に各種 CRUD 操作を実行するように 1 つのプロセスを設定することができます。XQuery 関数を使用して、要求で使用されている HTTP 動詞とリソースパスセグメントを判別できます。

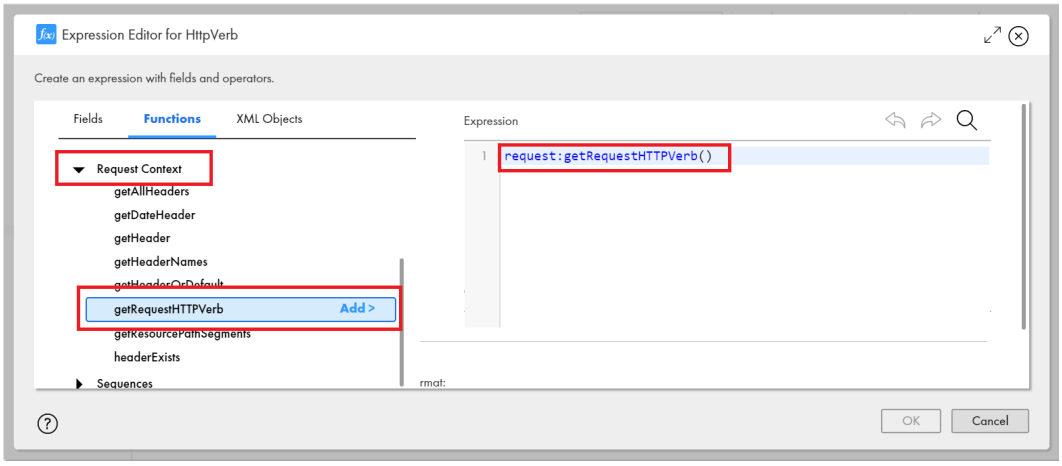
例えば、顧客レコードが格納された CRM システムを更新する場合に、次のタスクの 1 つ以上を実行する必要があるとします。

- GET 動詞を使用して顧客レコードを読み取る。
- POST 動詞を使用して新規顧客レコードを挿入する。
- PATCH 動詞を使用して既存の顧客レコードを更新する。
- PUT 動詞を使用して既存の顧客レコードを上書きする。
- DELETE 動詞を使用して既存の顧客レコードを削除する。

次の図に示すように、これらすべての操作を実行するように 1 つのプロセスを設定することができます。



プロセスの割り当てステップでは、getRequestHTTPVerb 関数を使用して、要求で使用されている HTTP 動詞を判別します。次の図は、式エディタの **[要求コンテキスト]** セクションにある getRequestHTTPVerb 関数を示しています。



ディシジョンステップは、使用されている HTTP 動詞を基に、各種 CRUD 操作を実行するためのさまざまなパスに分岐します。

さまざまな HTTP 動詞を使用してプロセスを呼び出す方法に関するビデオを表示するには、次のコミュニティの記事を参照してください。

https://www.youtube.com/watch?v=bQaQ_KmBO4o

HTTP 動詞の関数

要求で使用される HTTP 動詞とリソースパスセグメントを取得するには、HTTP 動詞の関数を使用します。HTTP 動詞の関数は、式エディタの［**要求コンテキスト**］セクションで利用できます。

次の HTTP 動詞の関数を使用できます。

getRequestHTTPVerb

要求で使用されている HTTP 動詞を判別するには、getRequestHTTPVerb 関数を使用します。この関数によって、要求から HTTP 動詞が取得され、次の応答のいずれかが文字列形式で返されます。

- GET
- POST
- PATCH
- PUT
- DELETE

getResourcePathSegments

REST 要求のすべてのまたは特定のリソースパスセグメントを取得するには、getResourcePathSegments 関数を使用します。この関数によって、トークンの文字列として値が返されます。

例えば、次の要求 URL があるとします。

https://na1.ai.dm-us.informaticacloud.com/active-bpel/rt/InitiateOrder/Orders/OrderID_100/quantity/5

request:getResourcePathSegments()式を使用すると、次の応答が返されます。

[Orders, OrderID_100, quantity, 5]

特定のリソースパスセグメントを取得するには、数値修飾子を使用して、プロセス名からのリソースパスセグメントの位置を示します。

例えば、OrderID_100 セグメントだけを応答で取得するには、次の式を使用します。

request:getResourcePathSegments()[2]

注: getResourcePathSegments 関数を使用して、SOAP 要求およびメッセージイベントのリソースパスセグメントを取得することはできません。getResourcePathSegments 関数を使用して、ホストコンテキストを取得することはできません。

REST エンドポイントとデータ変換

REST エンドポイント

プロセスを次の形式を使用する URL から REST サービスとして公開することができます。

https://mySPIwebAdress:portNumber/active-bpel/services/REST/<process_name>?var=value

次のいずれかの動詞を使ってプロセスを呼び出すことができます。

- GET
- POST
- PATCH

- PUT
- DELETE

POST 動詞を含んだ要求を送信する場合は、JSON を使用します。

例えば、次のスニペットは、POST 動詞を使用した JSON 要求を示しています。

```
{
        "param_A" : "abc", // NOTE: string, no $t
        "param_B" : 123,   // NOTE: typed as a number
        "param_C" : true   // NOTE: typed as bool
 }
```

要求によって次の応答が返されます。

```
{
        "field_A" : "abc",
        "field_B" : 123,
        "field_C" : true
 }
```

GET 動詞を含んだ要求を送信する場合は、クエリパラメータを使用します。

例えば、次のスニペットは、GET 動詞を含んだ同じ要求を示しています。

```
GET/rt/ProcessName?param_A=abc&param_B=123&param_C=true
```

単純型の配列の処理

単純型の配列が含まれた JSON ペイロードをプロセスオブジェクトに配置できます。例えば、ObjectList 型の orders という入力フィールドをプロセス内で定義することができます。orders 入力フィールドでは、1 つのフィールドを含んだ OrderNumber というプロセスオブジェクトを参照できます。

次のスニペットは、送信できる JSON 配列を示しています。

```
{
 "orders" : ["O-213", "O-425"]
}
```

Process Designer により、この配列は内部的に以下に変換されます。

```
{
 "orders" : [{OrderNumber : "O-213"},
             {OrderNumber : "O-425"}]
}
```

出力フィールドが JSON にシリアル化されると、Process Designer はオブジェクトを単純型の配列に変換します。

プロセスの Swagger JSON 仕様

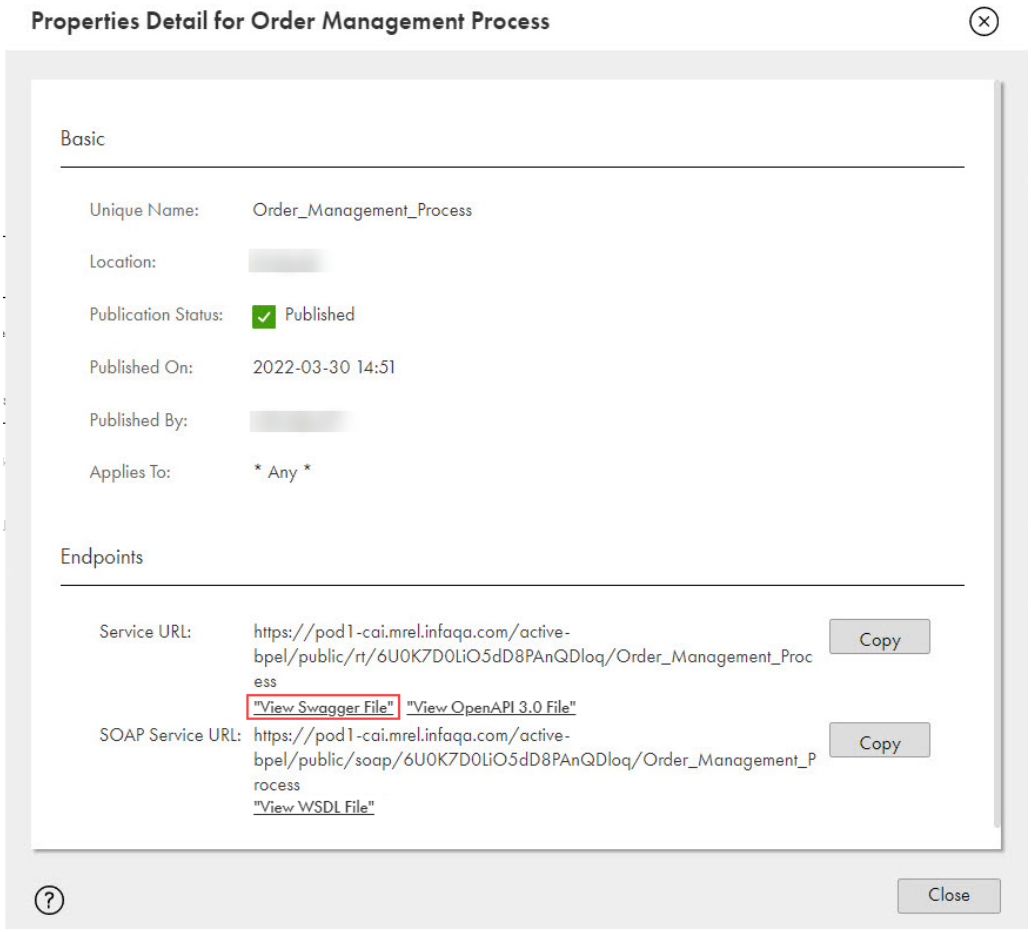
Informatica Process Designer で作成したプロセスのペイロードの Swagger JSON の仕様を表示できます。

プロセスの Swagger JSON 仕様には、入力、出力、プロセスオブジェクトなどのプロセスペイロード機能が含まれています。Swagger JSON 仕様には、version、info、host、basePath、schemes、paths の標準 Swagger プロパティ要件が含まれています。

プロセスの Swagger JSON 仕様へのアクセス

プロセスの Swagger JSON 記述にアクセスするには、次のタスクのいずれかを実行します。

- Swagger JSON 仕様を表示するプロセスを開き、**［アクション］** > **［プロパティの詳細］**をクリックします。次に、サービス URL の隣にある**［Swagger ファイルの表示］**をクリックします。
  次の図に、Swagger ファイルにアクセスするためのリンクを示します。



- プロセスを開き、サービス URL の隣にある**［アクション］** > **［プロパティの詳細］** > **［コピー］**をクリックします。次に、ブラウザウィンドウにサービス URL を貼り付け、サービス URL に?Swagger を追加します。
  例えば、サービス URL が `https://bs1e1.rt.informaticacloud.com/active-bpel/rt/sample_process` である場合、ブラウザウィンドウに `https://bs1e1.rt.informaticacloud.com/active-bpel/rt/sample_process?Swagger` を貼り付けます。

プロセスの Swagger JSON 仕様には、HTTP 応答ステータスコード、ステップ名、および終了ステップまたはマイルストーンステップで設定した HTTP 応答ヘッダーの名前とタイプが表示されます。ただし、プロセスに受信ステップが含まれる場合、HTTP 応答ステータスコードは設定に関係なく**［200 OK］**のままになります。アプリケーションの統合がサポートする応答ステータスコードの詳細については、「「HTTP 応答ステータスコード」 (ページ 31)」を参照してください。

プロセスの Swagger JSON 仕様の使用

プロセスの Swagger JSON 仕様をサードパーティツールとともに使用できます。例えば、プロセスの Swagger JSON 仕様を Swagger.io codegen ツール（http://swagger.io/swagger-codegen/）とともに使用できます。

内部のツールとともに Swagger JSON 記述を使用できます。例えば、プロセスの Swagger JSON 仕様を使用して、サービスコネクタをインポートするために使用するファイルを作成できます。詳細については、*設計 > サービスコネクタの作成 > サービスコネクタの生成*を参照してください。

プロセスの OpenAPI JSON 仕様

OpenAPI 仕様（以前は Swagger 仕様と呼ばれていました）は、REST API エンドポイントを記述するための形式です。

アプリケーション統合は、OpenAPI 仕様バージョン 3.0.1 をサポートします。Process Designer で作成したプロセスのペイロードの OpenAPI 仕様を表示できます。

プロセスの OpenAPI JSON 仕様には、入力、出力、説明、プロセスオブジェクトなどのプロセスペイロード情報が含まれています。OpenAPI 仕様では、`version`、`info`、`servers`、`paths`、`components` の標準 OpenAPI プロパティ要件を指定します。

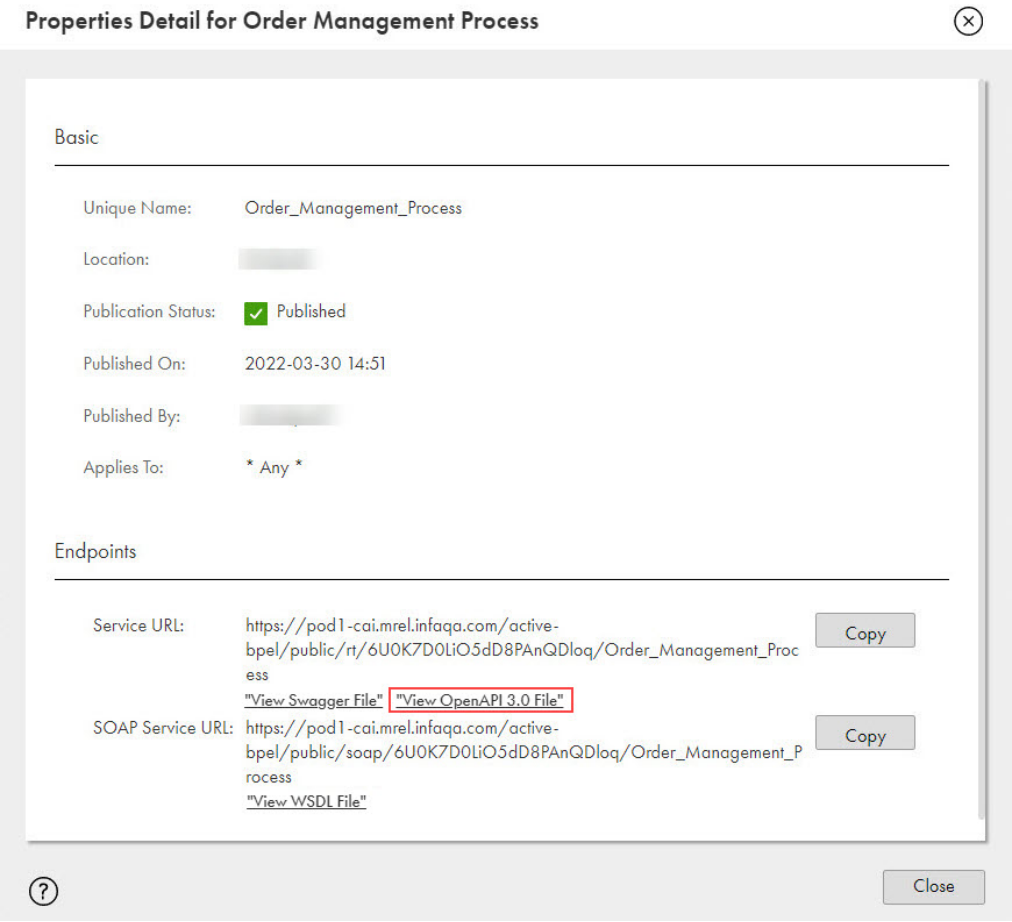
OpenAPI 3.0.1 仕様は Swagger 2.0 よりも説明的で、複雑なドキュメントを作成することができます。

プロセスの OpenAPI JSON 仕様へのアクセス

プロセスの OpenAPI 3.0 仕様にアクセスするには、次の手順を実行します。

1. **［参照］**ページで、OpenAPI 仕様を表示するプロセスに移動します。
2. **［アクション］**メニューで、**［プロパティの詳細］**をクリックして、生成された OpenAPI ファイルを表示します。
   ［プロパティの詳細］ダイアログボックスが表示されます。

   次の図に、OpenAPI 3.0 ファイルにアクセスするためのリンクを示します。



3. **［OpenAPI 3.0 ファイルの表示］**をクリックして、関連する OpenAPI ファイルを表示します。OpenAPI 3.0.1 仕様が新しいタブで開きます。

プロセスの OpenAPI 3.0.1 仕様には、HTTP 応答ステータスコード、ステップ名、および終了ステップまたはマイルストーンステップで設定した HTTP 応答ヘッダーの名前とタイプが表示されます。ただし、プロセスに受信ステップが含まれる場合、HTTP 応答ステータスコードは設定に関係なく **［200 OK］**のままになります。アプリケーションの統合がサポートする応答ステータスコードの詳細については、「HTTP 応答ステータスコード」 (ページ 31)」を参照してください。

プロセスの実行

プロセスをパブリッシュしたら、プロセスを実行してテストできます。プロセス入力を作成し、プロセス入力を使用してプロセスを実行できます。プロセスを実行した後、プロセスインスタンス実行の成功と失敗の詳細を表示できます。

次のビデオは、プロセス入力を作成し、プロセス入力を使用してプロセスをテストする方法を示しています。

https://knowledge.informatica.com/s/article/DOC-18509

プロセス入力の作成

プロセスをパブリッシュした後、プロセス入力を作成し、それを使用してプロセスをテスト目的で実行できます。また、複数のプロセス入力を作成し、すべての入力を使用してプロセスを実行することもできます。

プロセス入力は、JSON 形式または XML 形式で作成できます。次に、プロセス入力を検証して保存します。

プロセス入力を含むプロセスをエクスポート、コピー、または移動すると、プロセス入力も保持されます。

1. **[参照]** ページで、プロセス入力を作成するプロセスに移動します。
2. **[アクション]** メニューで、**[使用して実行]** をクリックします。

   [プロセス入力コレクションのテスト] ページが開きます。

   以下の図は、**[プロセス入力コレクションのテスト]** ページを示しています。

3. **[新しい入力]** をクリックします。
4. プロセス入力の名前を入力して、**[保存]** をクリックします。

   プロセス入力の名前は 80 文字を超えないようにする必要があります。

   [プロセス入力] セクションは、プロセスに必要な入力フィールドで読み込まれます。

   以下の図は、**[従業員]** プロセスオブジェクトの入力フィールドで読み込まれた **[プロセス入力]** セクションを示しています。

5. **[エンコーディング]** リストから、使用する形式に基づいて **[JSON]** または **[XML]** を選択し、プロセス入力を指定します。

   注: プロセスにスペース文字を含む入力フィールドが含まれている場合、エンコーディングタイプに **[JSON]** を選択します。
6. **[検証]** アイコンをクリックし、プロセス入力の構文を検証します。

   検証が成功したかどうかを示す確認メッセージが表示されます。検証が失敗した場合、プロセス入力の構文を修正し、再度検証します。
7. **[保存]** をクリックして、プロセス入力を保存します。

   [名前を付けて保存] をクリックして、プロセス入力を別の名前で保存します。**[リセット]** アイコンをクリックして、プロセス入力を最後に保存したプロセス入力にリセットすることもできます。

プロセス入力を作成した後、必要な入力を使用してプロセスを実行できます。

プロセス入力を使用したプロセスの実行

プロセスをパブリッシュし、プロセス入力を作成した後、プロセスをプロセス入力を使用して実行し、テストすることができます。プロセスを実行した後、プロセスインスタンス実行の成功と失敗の詳細を表示できます。

1. **［参照］**ページで、1 つ以上のプロセス入力を使用して実行するプロセスに移動します。
2. 次のいずれかの手順に従います。
   - 最後に実行が成功したプロセス入力を使用してプロセスを実行するには、**［アクション］**メニューを開いて、**［実行］**をクリックします。**［プロセス実行ステータス］**ページが開き、プロセス実行の成功と失敗の詳細が表示されます。

     注: プロセス入力を前に作成しなかった、または実行しなかった場合、**［プロセス入力コレクションのテスト］**ページが開き、そこでプロセス入力を作成し、プロセスを実行することができます。
   - 特定のプロセス入力またはすべてのプロセス入力を使用してプロセスを実行するには、**［アクション］**メニューを開き、**［使用して実行］**をクリックします。**［プロセス入力コレクションのテスト］**ページが開きます。

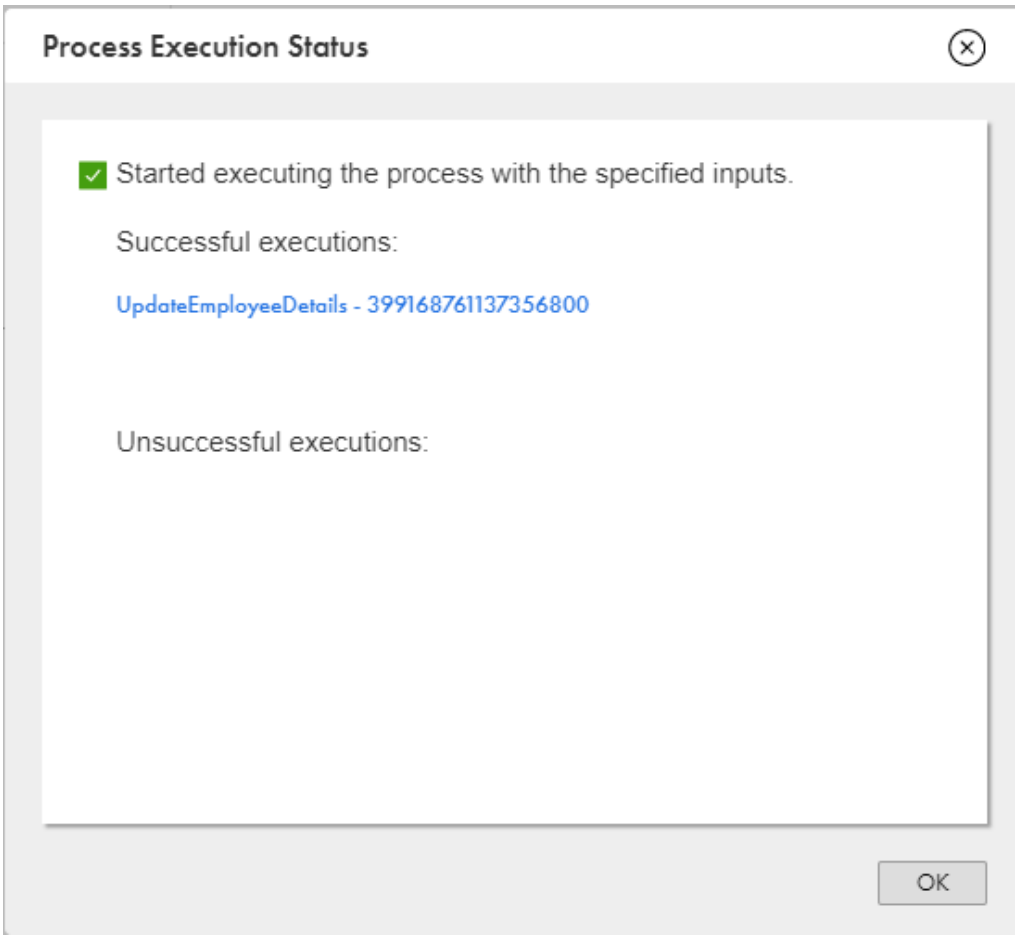
     以下の図は、**［プロセス入力コレクションのテスト］**ページを示しています。

注: プロセスに保存またはパブリッシュされていない変更が含まれている場合、**［実行］**オプションと**［使用して実行］**オプションは無効になります。プロセスを実行するには、プロセスを保存して再度パブリッシュする必要があります。

3. 次のいずれかの手順に従います。
   - 特定のプロセス入力を使用してプロセスを実行するには、プロセス入力を選択し、**［実行］**をクリックします。
   - すべてのプロセス入力を使用してプロセスを実行するには、**［すべて実行］**をクリックします。

   アプリケーション統合で指定された入力を使用してプロセスが実行されます。プロセス実行が完了すると、**［プロセス実行ステータス］**ページが開き、プロセス実行の成功と失敗の詳細が表示されます。プロセスの実行に時間がかかる場合、**［プロセス実行ステータス］**ページが表示されるまでに時間がかかることがあります。

   以下の図は、**［プロセス実行ステータス］**ページを示しています。



4. プロセス実行の詳細を表示するには、**[実行成功]** セクションまたは **[実行失敗]** セクションのリンクをクリックします。

   プロセス実行 ID がリンクに追加されます。実行 ID を使用して、アプリケーション統合コンソールでプロセス実行詳細を確認できます。

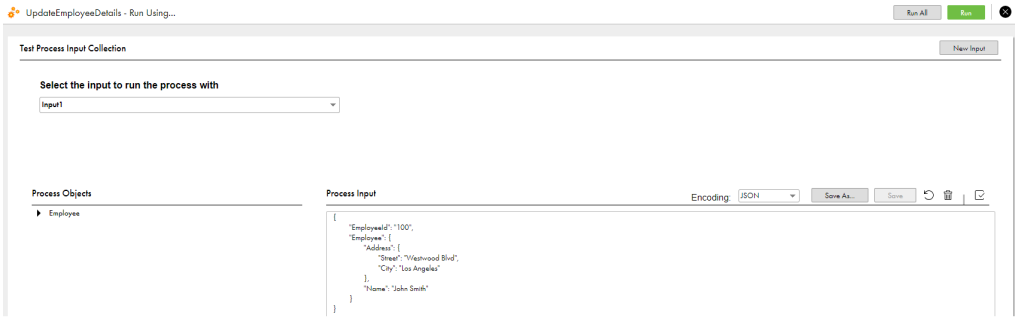
プロセス入力の削除

不要なプロセス入力は削除できます。

1. **[参照]** ページで、プロセス入力を削除するプロセスに移動します。
2. **[アクション]** メニューで、**[使用して実行]** をクリックします。

   [プロセス入力コレクションのテスト] ページが開きます。

   以下の図は、**[プロセス入力コレクションのテスト]** ページを示しています。



3. 削除するプロセス入力を選択し、**［削除］**アイコンをクリックします。

   プロセス入力の削除の確認を求めるメッセージが表示されます。

4. 削除を続行するには**［削除］**、削除をキャンセルするには**［キャンセル］**をクリックします。

プロセス入力のルールおよびガイドライン

プロセス入力を作成してプロセスを実行する場合、次のルールおよびガイドラインを考慮してください。

- プロセスにスペース文字を含む入力フィールドが含まれている場合、プロセス入力の作成時に XML エンコーディングタイプを使用することはできません。
- データ型が画像、添付、添付ファイルのプロセス入力は作成できません。
- データ型がリッチテキスト領域のプロセス入力を作成する場合、そのプロセス入力が適切にエンコードされていることを確認する必要があります。< >を< >.としてエンコードする必要があります
  テキスト入力フィールドの名前を太字の InputField、テキスト入力値を太字の HelloWorld とします。この例では、プロセス入力コードを次のように編集する必要があります。

  ```
  { "InputField": " &amp;lt;bold&amp;gt; HelloWorld &amp;lt;/bold&amp;gt; " }
  ```

- 任意データ型のプロセス入力を作成する場合、プロセス入力をキーと値のペアとして指定する必要があります。
  任意データ型の入力フィールドの名前を InputFieldAnyType、値を valueHelloWorld とします。この例では、JSON でエンコードされているプロセス入力コードは次のようである必要があります。

  ```
  {{ "InputFieldText": "HelloWorld", "InputFieldAnyType": { "key": "valueHelloWorld" }}
  ```

  エンコーディングタイプに**［XML］**を選択する場合、プロセス入力コードは次のようである必要があります。

  ```
  <root>
  <InputFieldText>HelloWorld</InputFieldText>
  <InputFieldAnyType>
  <key>HelloWorld</key>
  </InputFieldAnyType>
  </root>
  ```

プロセス呼び出しのアクティビティ実行制限の制約

10,000 のアクティビティ制限違反が発生してプロセスの呼び出しが失敗した場合、次の 15 分間はプロセスを実行できず、この期間中にプロセスデプロイメントの新しいインスタンスは作成されません。

適切な再試行メカニズムやループブレーカメカニズムを持たないプロセスを実行すると、次のエラーが発生します。

```
{"error":{"code":429,"message":"The last execution of this process has exceeded the activity execution limit at [07-Sep-2023 06:55:23 GMT]. As a result, new instances of this process deployment will not be created for a period of [15] minutes. You can review the process logs to find out the root cause of the activity limit breach."}}
```

プロセスログを確認して、アクティビティ制限違反の根本原因を見つけることができます。

15 分の制限時間が経過した後、またはプロセスを再パブリッシュすることによって、プロセスの実行を再開できます。

Secure Agent クラスタ上で無限ループに陥るプロセスをパブリッシュすると、動作の競合が発生し、プロセスを実行するたびにエラーメッセージが表示されない場合があります。これは、プロセスインスタンスの保存に使用されるメモリ内キャッシュがノード固有であり、Secure Agent クラスタ内のすべてのノードにブロードキャストされないためです。システムが自動的にノードを選択するため、マルチノードクラスタ設定では、あるノードではプロセスが拒否されても、別のノードでは実行される場合があります。各ノードは独自のキャッシュを維持するため、プロセスが実行されるノードに応じて、プロセスの失敗と実行可能な再試行を独立して処理できます。

例

プロセス A がノード 1 で実行中に、429 - 要求が多すぎますエラーが発生した場合。その結果、ノード 1 に固有のメモリ内キャッシュに、アクティビティ制限違反によってプロセス A が失敗したことを示すエントリが追加されます。ノード 1 上のプロセス A の新しいインスタンスは 15 分間作成されません。

ただし、プロセス A の実行がノード 2 にルーティングされる場合、ノード 2 はプロセスを再度実行しようとします。ノード 2 のプロセス A で同じ 429 - 要求が多すぎますエラーが発生した場合、新しいエントリがノード 2 の固有のキャッシュに追加されます。

注: プロセス呼び出し機能のアクティビティ実行制限の制約はテクニカルプレビュー段階です。テクニカルプレビュー機能はサポートされていますが、保証対象外で本番環境には対応していません。非本番環境でのみ使用することをお勧めします。Informatica では、本番環境用に次の GA リリースでプレビュー機能を導入するつもりですが、市場や技術的な状況の変化に応じて導入しない場合もあります。詳細については、Informatica グローバルカスタマサポートにお問い合わせください。

プロセスの終了

非永続プロセスが 3600 秒、つまり 1 時間を超えて実行されると、自動的にタイムアウトして終了します。タイムアウトにより、システムリソースの浪費を防ぎます。

プロセスの保持

プロセスは、Informatica によって設定された保持（パージ）ポリシーに従います。

自動データベースメンテナンスが有効になっている場合、すべてのプロセスに設定されているデフォルトの保持日数がプロセスに適用されます。デフォルトでは、完了したプロセスインスタンスはプロセスサーバーデータベースに 24 時間、つまり 1 日残ります。障害が発生したプロセスインスタンスは、プロセスサーバーデータベースに 72 時間、つまり 3 日間残ります。プロセスインスタンスは、保持期間が経過すると自動的に削除されます。サーバーログは、プロセスサーバーデータベースに 30 日間残ります。

アプリケーション統合コンソールで個々のプロセスに対して設定されたプロセスインスタンスの保持設定は、すべてのプロセスのデフォルトの保持日数設定よりも優先されます。

注: このトピックの保持期間に関する情報は、予告なく変更される場合があります。

プロセスインスタンスの保持の詳細については、「Process Instance Retention」を参照してください。

第 2 章

ガイドのランタイムタスク

ガイドを作成した後、以下のランタイムタスクを実行できます。

ガイドをパブリッシュする

ガイドを実行したり、ガイドをサードパーティアプリケーション内に組み込んだりするには、ガイドをパブリッシュする必要があります。

ガイドを実行する

ガイドをパブリッシュしたら、Guide Designer からガイドを実行してテストできます。

サードパーティアプリケーション内へのガイドの組み込み

ガイドをパブリッシュすると、アプリケーション統合は組み込みコードを生成します。その組み込みコードを使用して、サードパーティアプリケーション内にガイドを組み込むことができます。

ガイドのパブリッシュ

ガイドを実行したり、ガイドをサードパーティアプリケーション内に組み込んだりするには、ガイドをパブリッシュする必要があります。

複数のガイドをまとめてパブリッシュすることもできます。詳細については、第 3 章,「アプリケーション統合アセットの一括パブリッシュ」 (ページ 29)を参照してください。

パブリッシュされたガイドを編集する場合、変更を反映させるためにガイドを再度パブリッシュする必要があります。そうしなかった場合、**[プロパティの詳細]** ダイアログボックスと **[参照]** ページで、そのガイドのステータスが **[古くなっている]** に変わります。**[古くなっている]** ステータスは、ガイドにパブリッシュされていない変更が含まれているということです。

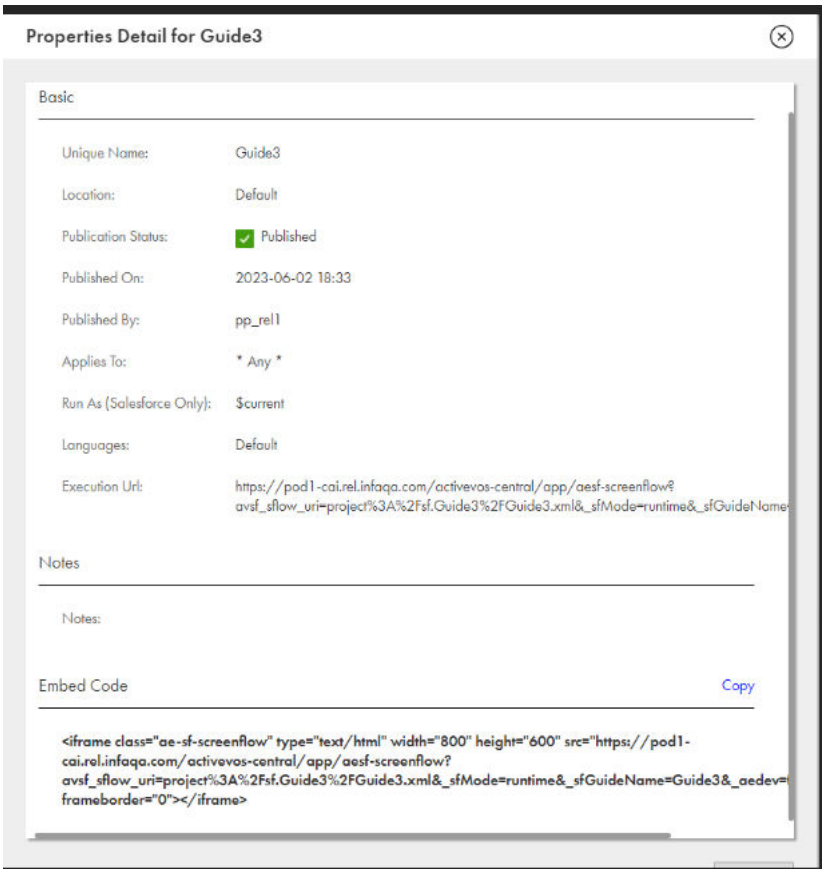
1. **[参照]** ページで、パブリッシュするガイドに移動し、**[パブリッシュ]** をクリックします。

   アプリケーション統合でガイドがパブリッシュされ、実行 URL と埋め込みコードが生成されます。

2. **[アクション]** メニューで、**[プロパティの詳細]** をクリックして、生成された実行 URL と組み込みコードを表示します。

   [プロパティの詳細] ダイアログボックスが表示されます。

   以下の図は、**[プロパティの詳細]** ダイアログボックスを示しています。



3. **［実行 URL］**フィールドでは、ガイドの実行に使用できる実行 URL を表示できます。
4. **［組み込みコード］**セクションで、**［コピー］**をクリックして組み込みコードをコピーします。

   その後、その組み込みコードを使用して、サードパーティアプリケーション内にガイドを組み込むことができます。

   注: サードパーティのアプリケーションが組み込みコードをサポートしていない場合は、実行 URL を使用してガイドを実行できます。

ガイドのパブリッシュ解除

ガイド名を編集したり、パブリッシュされたガイドを無効にしたりするには、ガイドをパブリッシュ解除する必要があります。パブリッシュされていないガイドをサードパーティアプリケーションから呼び出すことはできません。したがって、ガイドをパブリッシュ解除した後で、それに応じてサードパーティアプリケーションを更新する必要があります。

1. **［参照］**ページで、パブリッシュ解除するガイドに移動します。
2. **［アクション］**メニューで、**［パブリッシュ解除］**をクリックします。

   アプリケーション統合で、ガイドがパブリッシュ解除され、ガイドと組み込みコードが無効化されます。

必要な変更を行ったら、ガイドを再度パブリッシュして組み込みコードを生成し、ガイドを有効にします。

ガイドの実行

ガイドをパブリッシュしたら、ガイドを実行してテストできます。

ガイドが Salesforce オブジェクトに適用される場合、Salesforce でガイドを実行します。Salesforce でガイドを実行する方法の詳細については、『*Salesforce およびアプリケーション統合ガイド*』を参照してください。

Guide Designer からガイドを実行するには、次の手順を実行します。

1. **［参照］**ページで、実行するガイドに移動します。
2. **［アクション］**メニューで、**［実行］**をクリックします。

   注: ガイドに保存またはパブリッシュされていない変更が含まれている場合、**［実行］**オプションは無効になります。ガイドを実行するには、ガイドを保存して再度パブリッシュする必要があります。

   ガイドがブラウザで開きます。

サードパーティアプリケーション内へのガイドの組み込み

ガイドをパブリッシュすると、アプリケーション統合はガイドの組み込みコードを生成します。組み込みコードには、インラインフレームを指定する<iframe>タグが含まれています。インラインフレームを使用して、ガイドをサードパーティアプリケーションの HTML ドキュメントに組み込むことができます。

ガイドの終了

ユーザーがガイドに入力を行わず、ガイドを実行したままにすると、ガイドは非アクティブになります。非アクティブなガイドは、7 日後に自動的に終了します。非アクティブなガイドがマイルストーンステップにある場合、ガイドは 14 日後に自動的に終了します。

ガイド保持

ガイドは、Informatica によって設定された保持（パージ）ポリシーに従います。

自動データベースメンテナンスが有効になっている場合、すべてのガイドに設定されているデフォルトの保持日数がガイドに適用されます。デフォルトでは、完了したガイドインスタンスはプロセスサーバーデータベースに 24 時間、つまり 1 日残ります。障害が発生したガイドインスタンスは、プロセスサーバーデータベースに 72 時間、つまり 3 日間残ります。ガイドインスタンスは、保持期間が経過すると自動的に削除されます。サーバーログは、プロセスサーバーデータベースに 30 日間残ります。

注: このトピックの保持期間に関する情報は、予告なく変更される場合があります。

第 3 章

アプリケーション統合アセットの一括パブリッシュ

パブリッシュリソースを使用すると、単一のアプリケーション統合アセットまたは複数のアプリケーション統合アセットを同時にパブリッシュし、時間を節約できます。プロセス、ガイド、アプリケーション接続、サービスコネクタを一括でパブリッシュできます。

1. REST クライアントでは、次の URL の POST 要求を使用します。

   <Cloud アプリケーション統合 POD の URL>/active-bpel/asset/v1/publish

2. 次のヘッダーを追加します。

キー	値
承認	application/vnd.api+json
Content-Type	application/vnd.api+json
INFA-SESSION-ID	login リソースを使用して、セッション ID を取得します。ログインリソースの詳細については、データ統合のヘルプの「REST API リファレンス」を参照してください。

3. 本文で assetPaths 属性を使用して、パブリッシュするアプリケーション統合アセットの 1 つ以上の場所と名前を指定します。

   次の形式を使用します。

```
{

  "data": {
    "type": "publish",
    "attributes": {
      "assetPaths": [
                        "Explore/<location-of-the-process>/<name-of-the-process>.PROCESS.xml",
                        "Explore/<location-of-the-guide>/<name-of-the-guide>.GUIDE.xml",
                        "Explore/<location-of-the-appconnection>/<name-of-the-
appconnection>.AI_CONNECTION.xml",

"Explore/<location-of-the-serviceconnector>/<name-of-the-serviceconnector>.AI_SERVICE_CONNECTOR.xml"
                    ]
    }
  }
}
```

4. POST 要求を送信します。

   パブリッシュジョブ ID と、成功応答または失敗応答が表示されます。要求が失敗した場合、応答にはエラーの詳細も含まれます。

次のスニペットは、応答の例を示しています。

```
{
    "data": {
        "type": "publish",
        "id": "690465325825028096",
        "attributes": {
            "jobState": "NOT_STARTED",
            "jobStatusDetail": {},
            "startedBy": "autouser_pod1",
            "startDate": "2022-03-21T07:42:42.000+0000",
            "totalCount": 1,
            "processedCount": 0,
            "assetPaths": [
                "Explore/Default/BulkPublishProcessToPublishOnCloud.PROCESS.xml"
            ]
        }
    },
    "links": {
        "self": https://na1.ai.dm-us.informaticacloud.com/active-bpel/asset/v1/publish/690465325825028096,
        "status": https://na1.ai.dm-us.informaticacloud.com/active-bpel/asset/v1/publish/
690465325825028096/Status
    }
}
```

この例のパブリッシュジョブ ID は 690465325825028096 です。

5. パブリッシュのステータスとパブリッシュジョブに関する情報を表示するには、次の URL の GET 要求を使用します。

URL	**説明**
<Cloud アプリケーション統合 POD の URL>/active-bpel/asset/v1/publish/<publishId>/Status	パブリッシュのステータスを表示します。
<Cloud アプリケーション統合 POD の URL>/active-bpel/asset/v1/publish/<publishId>	パブリッシュ情報を表示します。

第 4 章

付録

HTTP 応答ステータスコード

HTTP 応答ステータスコードは、クライアントの要求に応答してサーバーによって生成される 3 桁のコードです。応答ステータスコードは、サーバーがクライアントの要求をどのように処理したかを伝えるための迅速かつ簡潔な手段として機能します。

次の表に、アプリケーションの統合がサポートする応答ステータスコードを示します。

応答ステータスコード	理由	説明
100	続行	要求の最初の部分が受信され、サーバーでは却下されていないことを示します。
101	プロトコルの切り替え	サーバーが切り替えるプロトコルを示します。
102	処理	サーバーが完全な要求を受け入れ、完了していないことを示します。
200	OK	要求が成功したことを示します。
201	作成済み	要求が実行され、1 つ以上の新しいリソースが作成されたことを示します。
202	承認済み	要求の処理が承認され、処理が完了していないことを示します。
203	信頼できない情報	要求が成功し、エンクローズされたペイロードが、変換プロキシによってオリジンサーバーの 200（OK）応答から変更されたことを示します。

応答ステータスコード	理由	説明
204	コンテンツがありません	サーバーが要求を正常に実行し、送信する追加のコンテンツが応答ペイロード本文にないことを示します。
205	コンテンツのリセット	サーバーが要求を実行し、ユーザーエージェントがドキュメントビューをオリジンサーバーから受信した元の状態にリセットすることを要求していることを示します。
206	部分的なコンテンツ	要求が成功し、要求の Range ヘッダーに記述されている要求されたデータ範囲が本文に含まれていることを示します。
207	複数ステータス	複数のステータスコードが適切である可能性がある状況で、複数のリソースに関する情報を伝えます。
208	すでに報告済み	分散オーサリングおよびバージョニング（DAV）バインディングのメンバーがマルチステータス応答の前の部分ですでに列挙されており、再度含まれないことを示します。
226	IM 使用済み	サーバーが特定のリソースに対するクライアントの要求を実行したことを示し、応答は現在のインスタンスに適用された 1 つ以上のインスタンス操作の結果を表します。
300	複数選択	要求に複数の可能な応答があることを示します。ユーザーエージェントまたはユーザーは、それらのいずれかを選択する必要があります。
301	完全に移動しました	ターゲットリソースに新しい永続 URI が割り当てられており、このリソースへの以降の参照では、エンクローズされた URI のいずれかを使用する必要があることを示します。

応答ステータスコード	理由	説明
302	次が見つかりました:	ターゲットリソースが一時的に別の URI に存在することを示します。
303	その他を表示	Location ヘッダーフィールドの URI で示されるように、サーバーがユーザーエージェントを別のリソースにリダイレクトしていることを示します。これは、元の要求に対する間接的な応答を提供することを目的としています。
304	変更されていません	条件付き GET または HEAD 要求が受信され、条件が false と評価されなかった場合に 200（OK）応答が返されたことを示します。
305	プロキシの使用	廃止されたステータスコードを示します。
306	（未使用）	以前のバージョンの HTTP/1.1 で定義され、現在は使用されていない予約済みコードを示します。
307	一時リダイレクト	ターゲットリソースが一時的に別の URI に存在し、ユーザーエージェントがその URI への自動リダイレクトを実行する場合、要求メソッドを変更してはならないことを示します。
308	永続リダイレクト	ターゲットリソースに新しい永続 URI が割り当てられており、このリソースへの以降の参照では、エンクローズされた URI のいずれかを使用する必要があることを示します。
400	不正な要求です	不正な要求構文、無効な要求メッセージフレーミング、不正な要求ルーティングなど、クライアントエラーとみなされる原因により、サーバーが要求を処理できないこと、または要求の処理を拒否していることを示します。

応答ステータスコード	理由	説明
401	未承認	ターゲットリソースの有効な認証資格情報が不足しているため、要求が適用されなかったことを示します。
402	必要な支払い	将来の使用のために予約されている非標準の応答ステータスコードを示します。
403	禁止されています	サーバーが要求を理解し、承認を拒否したことを示します。
404	見つかりません	オリジンサーバーがターゲットリソースの現在の表現を見つけられなかったか、存在の公開を拒否していることを示します。
405	メソッドが許可されていません	要求行で受信したメソッドがオリジンサーバーによって認識されており、ターゲットリソースによってサポートされていないことを示します。
406	許容されません	要求コンテンツネゴシエーションで受信したプロアクティブネゴシエーションヘッダーフィールドに従って、ターゲットリソースにユーザーエージェントで許容される現在の表現がなく、サーバーがデフォルトの表現の提供を拒否していることを示します。
407	プロキシ認証が必要です	401（未承認）に似ていますが、プロキシを使用するにはクライアント自体を認証する必要があることを示します。
408	要求タイムアウト	サーバーが、待機準備ができた時間内に完全な要求メッセージを受信しなかったことを示します。
409	名前の重複	ターゲットリソースの現在の状態と競合するため、要求を完了できなかったことを示します。

応答ステータスコード	理由	説明
410	存在しません	ターゲットリソースへのアクセスがオリジンサーバーで利用できなくなり、この状態が永続的になる可能性があることを示します。
411	必要な長さ	Content-Length が定義されていない要求の受け入れをサーバーが拒否していることを示します。
412	事前条件に失敗しました	要求のヘッダーフィールドに指定された 1 つ以上の条件が、サーバーでのテスト時に false と評価されたことを示します。
413	ペイロードが大きすぎます	要求のペイロードがサーバーが処理できる量よりも大きいため、サーバーが要求の処理を拒否していることを示します。
414	URI が長すぎます	要求のターゲットがサーバーが解釈できる長さよりも長いため、サーバーが要求の処理を拒否していることを示します。
415	サポートされていないメディアタイプです	ペイロードの形式がターゲットリソースのこのメソッドでサポートされていないため、オリジンサーバーが要求の処理を拒否していることを示します。
416	範囲を満たすことができません	要求の Range ヘッダーフィールド内の範囲が、選択されたリソースの現在の範囲と重複していないこと、または無効な範囲、あるいは小さい範囲や重複する範囲の過剰な要求により、要求された範囲のセットが却下されたことを示します。
417	予測に失敗しました	要求の Expect ヘッダーフィールドで指定された予測値が、少なくとも 1 つの受信サーバーで満たされなかったことを示します。

応答ステータスコード	理由	説明
422	処理できないエンティティ	サーバーが要求エンティティのコンテンツタイプを理解しており、要求エンティティの構文は正しくても、含まれている命令を処理できなかったことを示します。
423	ロック状態	メソッドのソースまたは宛先リソースがロックされていることを示します。
424	依存関係に失敗しました	要求されたアクションが失敗した別のアクションに依存していたために、リソース上でメソッドを実行できなかったことを示します。
426	アップグレードが必要です	サーバーが現在のプロトコルを使用した要求の処理を拒否しても、クライアントが別のプロトコルにアップグレードすると処理できる可能性があることを示します。
428	事前条件が必要です	サーバーで、要求が条件付きである必要があることを示します。これは、If-Match などの必須の前提条件ヘッダーが欠落していることを意味します。
429	要求が多すぎます	ユーザーが一定期間内に送信した要求が多すぎることを示します。
431	要求ヘッダーフィールドが大きすぎます	要求の HTTP ヘッダーが長すぎるため、サーバーが要求の処理を拒否したことを示します。
500	サーバーの内部エラー	サーバーで予期しない状態が発生したため、要求を完了できなかったことを示します。
501	実装されていません	サーバーが要求を実行するために必要な機能をサポートしていないことを示します。
502	不正なゲートウェイ	サーバーがゲートウェイまたはプロキシとして機能し、要求を実行しようとしたときに、アクセスした受信サーバーから無効な応答を受信したことを示します。

応答ステータスコード	理由	説明
503	サービスが使用不可能になりました	一時的な過負荷または計画されたメンテナンスのため、現在サーバーが要求を処理できないことを示します。ただし、これはしばらくすると解決される可能性があります。
504	ゲートウェイのタイムアウト	サーバーがゲートウェイまたはプロキシとして機能しているときに、要求を完了するためにアクセスする必要があるアップストリームのサーバーからタイムリーな応答を受信しなかったことを示します。
505	HTTP バージョンはサポートされていません	サーバーが要求メッセージで使用された HTTP のメジャーバージョンをサポートしていないか、サポートを拒否していることを示します。
506	バリアントもネゴシエートしています	これは、透過的なコンテンツネゴシエーションのコンテキストで発生する可能性があります。このプロトコルを使用すると、サーバーが複数のバリアントをサポートしている場合、クライアントは特定のリソースの最適なバリアントを取得できます。
507	ストレージが不足しています	サーバーが要求を完了するために必要な表現を格納できないことを示します。
508	ループが検出されました	`Depth: infinity` を使用した要求の処理中に無限ループが発生したため、サーバーが操作を終了したことを示します。

応答ステータスコード	理由	説明
510	拡張されません	要求がリソースへのアクセスに関するポリシーを満たさなかったことを示します。サーバーは、クライアントが拡張要求を発行するために必要なすべての情報を送り返す必要があります。拡張機能がクライアントにどのように通知するかという指定については、この仕様の範囲外です。
511	ネットワーク認証が必要です	ネットワークアクセスを取得するにはクライアントが認証する必要があることを示します。

この情報は RFC 7231 から参照されています。応答ステータスコードの詳細については、「RFC 7231- Response Status Codes」を参照してください。