ARIS PROCESS MINING

データ インジェスト API

バージョン 10.0 - SERVICE RELEASE 27 AND HIGHER

2024 年 10 月

This document applies to ARIS Process Mining Version 10.0 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

目次

1 パブリック API を使用するデータ インジェスト

ARIS Process Mining はパブリック データ インジェスト API をサポートします。API を使用するために HTTP リクエストを作成および送信します。API により、任意のデータ ソースから ARIS Process Mining にデータを転送できます。ARIS Process Mining に転送されたデータは論理テーブル構造であり、JSON 形式に適合している必要があります。

適切な API クライアントを使用して HTTP リクエストを作成できます。適切な API プログラミング スキルが必要です。

「データ転送オブジェクト (DTO)」『21ページ 』の章に、データ転送に使用できる DTO の一覧があります。

「API メソッド」『14ページ 』の章には、HTTP リクエストに使用できるエンドポイントの一覧があります。

ARIS PROCESS MINING での準備

API を使用してデータを ARIS Process Mining で転送するには、以下の手順を実行する必要があります。

- データ インジェスト API にシステム統合を作成します『2ページ 』。
- 転送されたデータを保存するデータ セットを作成します。
- データ インジェスト API への接続を作成します。『3ページ 』

データ インジェスト API を使用してデータを転送します。

以下の手順は、API を使用するデータ転送のベスト プラクティスです。

- API クライアントの認証『5ページ 』
- ソース テーブル定義の抽出『8ページ 』
- ソース テーブルの作成または置換『8ページ 』
- データ セットがデータをアップロードする準備を完了しているかを確認『10ページ 』
- データ アップロード サイクルの作成『11ページ 』
- データのアップロード『11ページ 』
- データ アップロード サイクルのコミット『12ページ 』
- サイクル状態の抽出『12ページ 』
- データ セットがデータを読み込む準備を完了しているかを確認『13ページ 』
- データ読み込みの開始『13ページ 』
- サイクル状態の抽出『14ページ 』

必要に応じて以下の手順を実行することもできます。

- ソース テーブルのドロップ『14ページ 』
- インジェスト サイクルの抽出『14ページ 』
- インジェスト サイクルのキャンセル『14ページ 』

1.1 注意

新しい JSON フィールドと列挙値は、API の軽微な変更の一環として Software GmbH によりインジェスト API 入力および出力に追加される場合があります。

1.2 ARIS Process Mining での準備

1.2.1 データ インジェスト API にシステム統合を作成する

データ インジェスト API を使用する『1ページ 』には、対応するシステム統合を作成する必要があります。

ARIS Process Mining は、認証方法として [クライアントの資格情報] 付与タイプと [認証コード] 付与タイプを使用して OAuth2 フローをサポートします。

クライアントの資格情報による認証情報はユーザーのコンテキスト外であり、マシン同士の通信に推奨されます。

必要条件

ARIS Process Mining Enterprise ライセンスがインストールされました。

手順

1. ⁝⁝⁝ [ナビゲーション メニュー] アイコン、プログラム ヘッダーの [管理] の順にクリックします。
2. [管理] パネルで [システム統合] をクリックします。
3. [システム統合の追加]、[データ インジェスト (API)] の順にクリックします。対応するダイアログ ボックスが開きます。
4. 「データ インジェスト」などの名前を入力し、オプションで説明を入力します。
5. [付与タイプ (OAuth)] ドロップダウン メニューで、認証方法を選択します。

   認証方法には [クライアントの資格情報] が推奨されます。これはマシン同士の通信に推奨され、実際のログオン ユーザーのコンテキストから外れます。

   [認証コード] 付与タイプを選択した場合、認証に使用される [認証コールバック URL] を指定します。

   https://<region.ariscloud>/umc/rest/oauth/callback?tenant=<project_room>&provider=umc

   ホスト名の <region.ariscloud> を ARIS Process Mining インストールのホスト名に、<project_room>をログオンする ARIS Process Mining プロジェクト ルームに置き換えます。

   ログオンしていれば、ブラウザーのアドレス バーで URL のホスト名 (例: processmining.ariscloud.com) を読み取ることができます。

   例

   ARIS Cloud 用の認証コールバック URL

https://processmining.ariscloud.com/umc/rest/oauth/callback?tenant=myprojectroom&provider=umc

ARIS Enterprise Cloud 用の認証コールバック URL

https://<my_company_name>.ariscloud.com/umc/rest/oauth/callback?tenant=<project_room>&provider=umc

6. [追加] をクリックします。[データ インジェスト アクセス データ] ダイアログ ボックスが開きます。ダイアログ ボックスに、クライアント ID、秘密鍵、周知の URL が表示されます。

   付与タイプ [認証コード] を選択した場合は、周知の URL が追加表示されます。

7. テキスト エディターを使用するなどして、提供された認証データを保存できます。

   [クリップボードへコピー] をクリックして、データを保存します。

8. [完了] をクリックします。

システム統合が作成され、指定した名前で一覧されます。

データ インジェスト API のシステム統合は、デフォルトでは [保留中] 状態で残りますので注意してください。ただし、システム統合は正常に使用できます。

ヒント

アクセス データ (エンドポイントを除く) は、作成したシステム統合に保存されます。クライアント資格情報キーにアクセスするために、ソース システム アクセス データを表示できます。

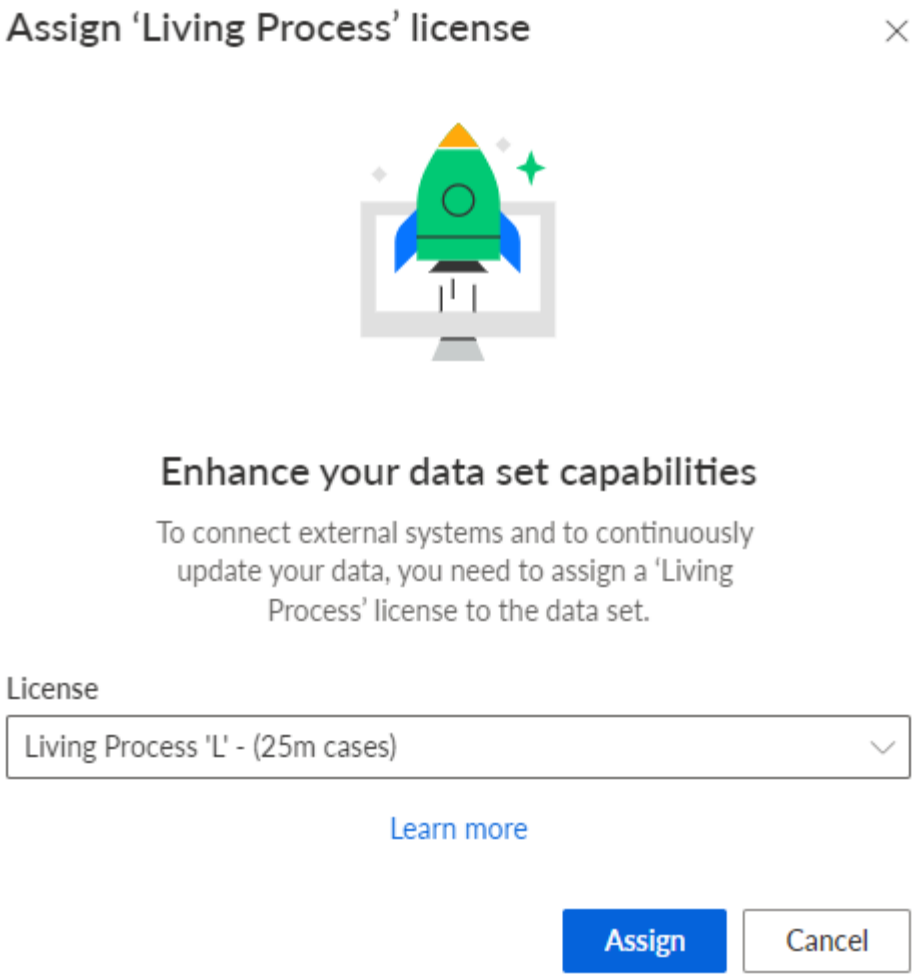
1.2.2 データ セットで接続を作成する

データ インジェスト API を使用してデータを ARIS Process Mining に転送するには、転送されたデータが保存されるデータ セットに対応する接続を作成する必要があります。作成したシステム統合 『2ページ 』を使用して、API クライアントへの接続を作成します。

手順

1. プログラム ヘッダーの ⁝⁝ [ナビゲーション メニュー] アイコンをクリックします。
2. ⛁ [データ コレクション] を選択します。[データ コレクション] が開き、[データ セット] ページが表示されます。
   a. データ セットを開いている場合は、最近開いたデータ セットが開きます。[データ セット] パネルの [戻る] をクリックし、[データ セット] ページを開きます。
   b. [データ セット] ページでデータ セットをクリックます。選択したデータ セットが開きます。
3. [接続] コンポーネントを開きます。
4. [接続の追加] をクリックします。ソース システムへの接続の追加が初めてであり、「Living Process」ライセンスをまだデータ セットに割り当てていない場合は、[「Living Process」ライセンスの割り当て] ダイアログ ボックスが開きます。

5. ドロップダウン メニューでライセンスを選択します。プロセスの抽出と分析には、「Living Process」ライセンスが必要です。抽出できるプロセスの数は、選択されているライセンスによって変わります。



6. [割り当て] をクリックします。[接続の追加] ダイアログ ボックスが開きます。
7. 接続を設定します。
   a. データ インジェストなどのソース システムへの接続に一意の名前を入力します。
   b. データ インジェスト API 用に作成されたシステム統合を選択します。
   c. [追加] をクリックします。

API への接続を作成しました。作成した接続は、指定した設定とともに [接続] ページに表示されます。

1.3 データ インジェスト API の使用

以下の手順は、データ インジェスト API を使用するデータ転送のベスト プラクティスです。

以下で記載されている操作は、ARIS Process Mining 用 webMethods.io コネクタを使用する『31ページ』際にあらかじめ定義された操作として使用できます。

1.3.1 API クライアントの認証

ARIS Process Mining に対してクライアントを認証するには、HTTP 認証リクエストを実行する必要があります。指定された認証方法『2ページ 』に応じて、クライアントの資格情報または認証コードを使用できます。

必要なデータは、データ インジェスト API 用に作成されたシステム統合『2ページ 』にあります。

クライアントの資格情報を使用する ARIS CLOUD に対する認証

クライアントの資格情報を使用する認証を強く推奨します。

mc.ariscloud.com という URL を使用してプロジェクト ルームにログオンする場合、ARIS Cloud を使用しています。

HTTP Post リクエストを ARIS Cloud エンドポイントと /api/applications/login (for example, https://mc.ariscloud.com/api/applications/login) パスに以下のプロパティとともに送信します:

- 内容タイプ: application/x-www-form-urlencoded
- リクエスト ボディと対応するシステム統合『2ページ 』からの値:

  clientId: クライアント ID

  clientSecret: クライアント シークレット

  tenant: プロジェクト ルーム名

応答は、テナント、URL、アクセス トークンで構成される JSON オブジェクトです。

```
{
"tenant": "<project_room>",
"token": "…",
"url": "https://some_url"
}
```

注意

REST エンドポイントへのすべての後続の呼び出し用のホスト名として、URL を使用します。

生成されたベアラー トークンが、すべての後続リクエストに適切なヘッダーとともに送信されることを確認します。そのためには、この HTTP リクエスト ヘッダーを各リクエストに以下のように追加します。

`Authorization: Bearer <応答からのトークン>`
用語の「Bearer」の後にある空白スペースに注意してください。

クライアントの資格情報を使用する、ARIS ユーザー管理による ARIS ENTERPRISE CLOUD に対する認証

クライアントの資格情報を使用する認証を強く推奨します。

HTTP Post リクエストを /umc/api/oauth/apptoke (例: https://my_company_name.ariscloud.com/umc/api/oauth/apptoken) パスと以下のプロパティとともに ARIS ユーザー管理に送信します:

- 内容タイプ: application/x-www-form-urlencoded

- リクエスト ボディと対応するシステム統合『2ページ 』からの値:

  client_id: クライアント ID

  client_secret: クライアント シークレット

  tenant: プロジェクト ルーム名

  grant_type: client_credentials

応答はアプリケーション トークンで構成される JSON オブジェクトです。

```
{
"applicationToken": "…"
}
```

生成されたベアラー トークンが、すべての後続リクエストに適切なヘッダーとともに送信されることを確認します。そのためには、この HTTP リクエスト ヘッダーを各リクエストに以下のように追加します。

```
Authorization: Bearer <応答からのトークン>
```

用語の「Bearer」の後にある空白スペースに注意してください。

認証コードを使用する認証

クライアント アプリケーションは [認証コード] 付与タイプを使用して OAuth 2.0 をサポートする必要があります。

クライアント アプリケーションが以下を使用するように設定します。

- **コールバック URL**

  ARIS Process Mining のプロジェクト ルームを認証するためにリダイレクトするコールバック URL。

  https://<region.ariscloud>/umc/rest/oauth/callback?tenant=<project_room>&provider=umc

  ログオンしていれば、ブラウザーのアドレス バーで URL のホスト名 (例: processmining.ariscloud.com) を読み取ることができます。

  例

  プロジェクト ルームが ARIS Cloud であれば、コールバック URL は次のようになります。

  https://processmining.ariscloud.com/umc/rest/oauth/callback?tenant=<project_room>&provider=umc

  プロジェクト ルームが ARIS Enterprise Cloud であれば、URL は次のようになります。

  https://<my_company_name>.ariscloud.com/umc/rest/oauth/callback?tenant=<project_room>&provider=umc

- **クライアント ID とクライアント シークレット**

  これらは ARIS Process Mining でシステム統合を作成した場合に分かり、また、システム統合用のシステム アクセス データを表示する際に ARIS Process Mining 管理の [システム統合] モジュールにある一覧から抽出できます。

  クライアント ID とシークレットは、基本 OAuth ヘッダーではなく、本文にて送信する必要があります。

- [認証、トークン、終了ポイントの更新] は、ブラウザーで対応する周知の URL を呼び出して抽出できます。システム統合用のシステム アクセス データを表示する際に、ARIS Process Mining 管理の [システム統合] モジュールで、一覧から周知の URL を抽出できます。URL は、authorization_endpoint、token_endpoint、refresh_endpoint、userinfo_endpoint とともに JSON オブジェクトを返します。

  例

  ```
  {
  "authorization_endpoint":
  "https://<hostname>/umc/oauthLogin?grant_type=authorization_code&tenant=<project_room>",
  "token_endpoint":
  "https://<hostname>/umc/api/v1/oauth/accesstoken?grant_type=authorization_code&tenant=<project_room>",
  "userinfo_endpoint":
  "https://<hostname>/umc/api/v1/oauth/userinfo?tenant=<project_room>",
  "refresh_endpoint":
  "https://<hostname>/umc/api/v1/oauth/refreshtoken?tenant=<project_room>"}
  }
  ```

クライアント アプリケーションが、これらのプロパティを使用して承認リクエストを送信した後に、サーバーがテナント、URL、アクセス トークンで構成される JSON オブジェクトで応答します。

```
{
"tenant": "<project_room>",
"token": "…",
"url": "https://some_url"
}
```

注意

REST エンドポイントへのすべての後続の呼び出し用のホスト名として、URL を使用します。

生成されたベアラー トークンが、すべての後続リクエストに適切なヘッダーとともに送信されることを確認します。そのためには、この HTTP リクエスト ヘッダーを各リクエストに以下のように追加します。

`Authorization: Bearer <応答からのトークン>`

用語の「Bearer」の後にある空白スペースに注意してください。

さらに、[認証コード] 認証タイプ用に CSRF トークンが各リクエストとともに送信される必要があります。

/umc/api/v2/tokens/csrftoken パスにある ARIS ユーザー管理に HTTP POST リクエストを送信して、認証が成功した後に CSRF トークンを取得できます。

結果は、現在のユーザー セッションに基づく英数字の文字列になります (例: oehltw0drUujSdWMD5TJEsXSLklwk1xKYh1LHaZ16g7)。その後の各リクエストは、このトークンを csrftoken ヘッダーとともに送信する必要があります。

1.3.1.1 URL パラメーターによる認証は非推奨です

サービス リリース 28 (SR 28) から、URL クエリ パラメーターを使用して認証資格情報を渡すことはサポートされなくなりましたので注意してください。資格情報はリクエスト ボディで渡される必要があります。『5ページ 』

1.3.1.2 データ セットの技術キーの意味

その後のすべてのセクションは、特定のデータ セットのコンテキストで実行されるさまざまなリクエストを表します。ほとんどすべてのリクエストに、データ セットの技術キーを使用する必要があります。『15ページ 』

技術キーを含むリクエストは次のようになります:

/dataSets/<データ セット>

<データ セット> パラメーターはデータ セットの技術キーを参照し、実行時にその値を使用します。値は、対応するデータ セットを開く際にブラウザーのアドレス バーにある URL から取得できます。

URL の形式は次のとおりです。

https://<ホスト名>/#<project_room>/dataCollection/y.dataset.<キー>

<キー> パラメーターは、データ セットの選択されている表示名に基づき、読取が可能なはずです。この特定のデータ セットに対するすべての API リクエストで、このキーを使用します。

例

https://ariscloud.com/#myprojectroom/dataCollection/y.dataset.mydataset

1.3.2 ソース テーブル定義の抽出

ソース テーブル定義を抽出するには、以下の HTTP リクエストを実行します。

GET "https://<ホスト名>/mining/api/pub/dataIngestion/v1/dataSets/<dataset>/sourceTableDefinitions[?fullyQualifiedNames=default.table_a[,default.table_b]]"

fullyQualifiedNames が指定されていない場合は、使用可能なすべてのソース テーブルの構造が返ります。

1.3.3 ソース テーブルの作成または置換

ソース テーブルを作成または置換するには、以下の HTTP リクエストを実行します。

ソース テーブルの作成

POST "https://<ホスト名>/mining/api/pub/dataIngestion/v1/dataSets/<データ セット>/sourceTables"

リクエスト ボディを以下の形式でサーバーに送信する必要があります (ここにあるのはサンプル データです):

```
[
  {
      "name": "table_a",
      "namespace": "default",
      "columns": [
        {
          "dataType": "STRING",
          "name": "column_a1"
        },
        {
          "dataType": "LONG",
          "name": "column_a2"
        },
        {
          "dataType": "DOUBLE",
          "name": "column_a3"
        },
        {
          "dataType": "FORMATTED_TIMESTAMP",
          "name": "column_a4",
          "format": "yyyy-MM-dd HH:mm:ss.SSS"
        }
      ]
  },
  {
      "name": "table_b",
      "namespace": "default",
      "persistenceMode": "OVERWRITE",
      "columns": [
       ...
      ]
  }
]
```

ソース テーブルの置換

POST "https://<ホスト名>/mining/api/pub/dataIngestion/v1/dataSets/<データ セット >/sourceTables?forceReplace=true"

リクエスト ボディを以下の形式でサーバーに送信する必要があります (ここにあるのはサンプル データです):

```
[
  {
      "fullyQualifiedName": "default.table_a",
      "columns": [
        {
          "dataType": "STRING",
          "name": "column_a1"
        },
        {
          "dataType": "LONG",
          "name": "column_a2"
        },
        {
          "dataType": "DOUBLE",
```

```
        "name": "column_a3"
      },
      {
        "dataType": "FORMATTED_TIMESTAMP",
        "name": "column_a4",
        "format": "yyyy-MM-dd HH:mm:ss.SSS"
      }
    ]
  },
  {
    "fullyQualifiedName": "default.table_b",
    "persistenceMode": "OVERWRITE"
  }
]
```

永続モード『29ページ 』(persistenceMode) が設定されていない場合は、persistenceMode = OVERWRITE を使用してテーブルが作成または置換されます。

ARIS Process Mining 用 WebMethods コネクタを使用している場合は、forceReplace パラメーターが暗黙的に設定されます。

1.3.4 データ セットがデータをアップロードする準備を完了しているかを確認

データ セットがデータをアップロードする準備を完了している必要があります。データ セットの状態を確認するには、以下の HTTP リクエストを実行します。

POST "https://<ホスト名>/mining/api/pub/dataIngestion/v1/dataSets/<データ セット >/readyForIngestion"

リクエスト ボディを以下の形式でサーバーに送信する必要があります (ここにあるのはサンプル データです):

```
{
  "dataUploadTargets": [
    {
      "fullyQualifiedName": "default.table_a"
    },
    {
      "fullyQualifiedName": "default.table_b"
    }
  ]
}
```

データ セットの準備が完了している場合は、肯定応答を受信します。そうでない場合は、該当する理由を含む否定応答が届きます。

例

```
{
  "ready": false,
  "cause": {
    "code": "INR1001",
    "message": "The data set is currently being processed"
  }
}
```

1.3.5 データ アップロード サイクルの作成

データ セットの準備が完了すると、以下の HTTP リクエストを使用してデータアップロード用のデータ インジェスト サイクルを作成できます。

POST "https://<ホスト名>/mining/api/pub/dataIngestion/v1/dataSets/<データ セット >/ingestionCycles"

リクエスト ボディを以下の形式でサーバーに送信する必要があります (ここにあるのはサンプル データです):

```
{
  "dataUploadTargets": [
    {
      "fullyQualifiedName": "default.table_a"
    },
    {
      "fullyQualifiedName": "default.table_b"
    }
  ]
}
```

上記の呼び出しに対する応答は、技術キー、参照されたデータ アップロード ターゲット (テーブル)、いくつかの状態情報を含む完全な形式のデータ インジェスト サイクルを返します。その初期状態は ACCEPTING_DATA です。サイクルによって参照されるすべてのテーブルは、その後のデータ アップロード以外のすべてに関してロックされます。

```
{
    "key": "data_set_1_55",
    "dataUploadTargets": [...],
    "dataLoadTriggered": false,
    "state": {
        "value": "ACCEPTING_DATA"
    }
}
```

サイクルの技術キーをメモして、そのサイクルのコンテキストで実行されるすべての後続リクエストに使用してください。たとえば、データ アップロード サイクルのコミットする際などにです。

技術キーを含むリクエストは次のようになります:

/ingestionCycles/<インジェスト サイクル>

1.3.6 データのアップロード

ソース テーブルのデータをデータ セットにアップロードするには、以下の HTTP リクエストを実行します。

POST "https://<ホスト名>/mining/api/pub/dataIngestion/v1/dataSets/<データ セット>/sourceTables/<ソース テーブル>/data"

<ソース テーブル>: 完全修飾名 (default.table_a)

リクエスト ボディを以下の形式でサーバーに送信する必要があります (ここにあるのはサンプル データです):

```
[
  ["This is a description", 1255, 1385.5, "2021-07-15 18:03:25.889"],
  ["A second example text", 510, -23.58, "2021-07-10 10:59:05.421"],
  ["Example text", 1626347163123, 3.1415, "2021-07-01 08:00:01.002"]
]
```

大量のデータは、複数回のリクエストを利用してアップロードできます。各リクエストごとに、データがサーバーに一時的な形式で保存されます。

1.3.7 データ アップロード サイクルのコミット

すべてのデータがアップロードされ、ソース テーブルに読み込ませることができることを ARIS Process Mining に示します。

PUT "https://<ホスト名>/mining/api/pub/dataIngestion/v1/dataSets/<データ セット>/ingestionCycles/<インジェスト サイクル>/dataComplete"

インジェスト サイクル状態が INGESTING_DATA に変わります。アップロードされた一時データは、この時点でソース データベースに残ります。

ARIS Process Mining でアップロードがエラーなしで完了すると、指定されているサイクルの状態が COMPLETED_SUCCESSFULLY に設定されます。

1.3.8 サイクル状態の抽出

サイクル状態を読み取るには、以下の HTTP リクエストを実行します。

GET "https://<ホスト名>/mining/api/pub/dataIngestion/v1/dataSets/<データ セット>/ingestionCycles/<インジェスト サイクル>/state"

データ インジェストにエラーがある場合は、該当する理由を含む応答を受信します。

```
{
  "value": "FAILED",
  "cause": {
```

```
    "code": "IER1000",
    "message": "An unexpected error occurred"
  }
}
```

それ以外の場合のインジェスト状態は、サイクルが依然として実行中の場合は INGESTING_DATA、問題が発生せずに完了した場合は COMPLETED_SUCCESSFULLY、中断された場合（API を使用した場合など）は CANCELED になります。

1.3.9 データ セットがデータを読み込む準備を完了しているかを確認

データ セットは、プロセス保管にデータを読み込ませる準備が完了している必要があります。データ セットの状態を確認するには、以下の HTTP リクエストを実行します。

POST "https://<ホスト名>/mining/api/pub/dataIngestion/v1/dataSets/<データ セット>/readyForIngestion"

リクエスト ボディを以下の形式でサーバーに送信する必要があります:

```
{
  "dataLoadTriggered": true
}
```

データ セットの準備が完了している場合は、肯定応答を受信します。そうでない場合は、該当する理由を含む否定応答が届きます。

1.3.10 データ読み込みの開始

データ セットがデータを読み込む準備を完了している場合は、以下の HTTP リクエストを使用してデータの読み込みを開始するデータ インジェスト サイクルを作成します。

POST "https://<ホスト名>/mining/api/pub/dataIngestion/v1/dataSets/<データ セット>/ingestionCycles"

リクエスト ボディを以下の形式でサーバーに送信する必要があります:

```
{
  "dataLoadTriggered": true
}
```

上記の呼び出しに対する応答は、完全な形式のデータ インジェスト サイクルを返します。初期状態は INGESTING_DATA です。対応するデータの読み込みがただちに開始されます。

ARIS Process Mining 用 WebMethods コネクタを使用している場合は、dataLoadTriggered パラメーターが暗黙的に設定されるので注意してください。

1.3.11 サイクル状態の抽出

データが無事に読み込まれたことを検証するには、以下の HTTP リクエストを実行します。

GET "https://<ホスト名>/mining/api/pub/dataIngestion/v1/dataSets/<データ セット>/ingestionCycles/<インジェスト サイクル>/state"

現在の状態値（INGESTING_DATA、COMPLETED_SUCCESSFULLY、または FAILED）を含む応答を受信します。オプションで、それぞれの理由を受信する場合もあります。

詳細については、「サイクル状態の抽出」『12ページ 』の章を参照してください。

1.3.12 ソース テーブルのドロップ

指定されているソース テーブルをドロップします。

DELETE "https://<ホスト名>/mining/api/pub/dataIngestion/v1/dataSets/<データ セット>/sourceTables/<ソース テーブル>"

1.3.13 インジェスト サイクルの抽出

すべての既存インジェスト サイクルを抽出します。

GET "https://<ホスト名>/mining/api/pub/dataIngestion/v1/dataSets/<データ セット>/ingestionCycles"

1.3.14 インジェスト サイクルのキャンセル

指定されているサイクルをキャンセルします。

PUT "https://<ホスト名>/mining/api/pub/dataIngestion/v1/dataSets/<データ セット>/ingestionCycles/<インジェスト サイクル>/canceled"

1.4 API メソッド

HTTP リクエスト用データ インジェスト API によって提供される以下のエンドポイントを使用できます。

1.4.1 パスのセクション: データ セット

API バージョン エンドポイント以外のすべてのエンドポイントは、特定のデータ セットのコンテキストで使用されます。データ セットのコンテキストがあるすべてのエンドポイントには、以下の URL セクションが含まれます:

/dataSets/{dataSet}

{dataSet} パラメーターはデータ セットの技術キーを参照し、実行時にその値を使用します。値は、対応するデータ セットを開く際にブラウザーのアドレス バーにある URL から取得できます。

URL の形式は次のとおりです。

https://<ホスト名>/#<project_room>/dataCollection/y.dataset.<キー>

<キー> パラメーターは、データ セットの選択されている表示名に基づき、読取が可能なはずです。この特定のデータ セットに対するすべての API リクエストで、このキーを使用します。

例

https://ariscloud.com/#myprojectroom/dataCollection/y.dataset.mydataset

1.4.2 パスのセクション: インジェスト サイクル

複数の終了ポイントが、データ アップロード サイクルのコミット『20ページ 』、インジェスト サイクル状態を返す『20ページ 』、インジェスト サイクルのキャンセル『19ページ 』などの特定のインジェスト サイクルのコンテキストで使用されています。そのため、それらのすべてに以下のセクションが含まれます:

/ingestionCycles/{ingestionCycle}

{ingestionCycle} パラメーターは、インジェスト サイクルの技術キーを参照し、実行時にその値を使用します。そのようなキーが、新しいインジェスト サイクルが作成されるたびに生成されます。これは、作成後、またはすべての既存のインジェスト サイクルが API を使用するシステムから抽出される場合に、サイクルのキー プロパティの値として返されます。

1.4.3 API バージョンの抽出

ARIS Process Mining の現在の API バージョンを抽出します。

GET /api/pub/dataIngestion/version

出力: 現在の API バージョン

1.4.4 ソース テーブル定義の抽出

指定されているソース テーブルの列構造（名前、データ型、形式）を抽出します。

GET /api/pub/dataIngestion/v1/dataSets/{dataSet}/sourceTableDefinitions

入力: 完全修飾名でフィルターを適用するための fqns クエリ パラメーター

出力: [SourceTableDefinition] オブジェクトの一覧

1.4.5 ソース テーブルの作成または置換

ARIS Process Mining でソース テーブルを作成または置換します。

これは forceReplace パラメーターに依存します。パラメーターが forceReplace = true の場合、同じ識別子を持つテーブルが置換され、それまでに保存されていたすべてのデータが削除されます。

POST /api/pub/dataIngestion/v1/dataSets/{dataSet}/sourceTables

入力:

- [SourceTableDefinition] オブジェクトの一覧
  - ソース テーブルを作成する際に、名前、名前空間、列が必須です。他のプロパティはオプションです。

    「_ARIS」は名前空間として許可されないので注意してください。
  - ソース テーブルを置換する際の注意事項

    - 既存テーブルの識別子は、キーと完全修飾名、または名前と名前空間のいずれかの形式で指定します。複数の識別子を指定する場合の優先順位は、キー、完全修飾名、名前と名前空間の順です。優先順位の低い識別子は、高い順位の識別子がある場合に無視されます。

    - その他のすべてのプロパティ（識別子は除く）はオプションです。プロパティが設定されていない場合は、既存テーブルの値が再使用されます。テーブルの列は同じプロパティに設定されますので注意してください。列を削除するには、本体から列を除外します。列を追加するには、既存列の情報を繰り返して、新しい列に含めます。

    「_ARIS」は名前空間として許可されないので注意してください。

- forceReplace クエリ パラメーターは、既存のソース テーブルを同じ識別子で置換するかどうかを示します。テーブルが置換される際に、それまでに保存されていたすべてのデータが削除されます。パラメーターに true が設定されていない場合は、置換（既存テーブルの識別子を含めるなど）が必要なリクエストは拒否されます。

出力:

新しく作成された、または置換されたソース テーブルに基づく [SourceTableDefinition] オブジェクトの一覧

1.4.6 ソース テーブル定義の更新

ソース テーブル定義を更新しますが、既存のデータはそのまま変更されずに残ります。これは以下のために使用できます。

- 完全修飾名、名前空間、名前のいずれかまたは全部の変更。

  「_ARIS」は名前空間として許可されないので注意してください。
- 標準テーブルからテーブル タイプを増分に変更。
- 増分テーブル用のマージ キーの（再）定義。
- 列を増分テーブルに追加します。
- 後続データの配信スキーマを、特定の列が含まれないように設定します。既存のデータは、列に含まれたままになります。

  標準テーブルの列は、この方法では変更できません。代わりにソース テーブルの終了ポイントを作成または置換『16ページ 』します。

POST /api/pub/dataIngestion/v1/dataSets/{dataSet}/sourceTables/{sourceTable}/definition

入力:

[SourceTableDefinition] オブジェクト

- ソース テーブルが既に存在する必要があります。
  - 既存のテーブルは保持されます。
  - そのため、定義に対するすべての変更が許可されるわけではありません。
  - URL にあるソース テーブルは、キーまたは完全修飾名のいずれかになります。
  - ソース テーブル定義のすべてのプロパティは任意のオプションです。プロパティが設定されていない場合は、既存テーブルの値が再使用されます。

    以下にご注意ください。

    テーブルの列はまとめて設定されます。つまり、列の削除は本文から削除するだけです。列を追加するには、既存列の情報を繰り返して、新しい列に含めます。

    本文に完全修飾名と名前の両方、または名前空間のいずれか、あるいは全部が提供されている場合、完全修飾名が優先されます。

出力:

新しく作成された、または置換されたソース テーブルに基づく [SourceTableDefinition] オブジェクトの一覧

増分テーブルのデータの更新は、ソース テーブルのスキーマへの変更も含む可能性があるので注意してください。

1.4.7 データ セットがインジェストの準備を完了しているかどうかを確認する

これは、データのアップロードまたはデータ読み込みを開始できることを確認します。

データ読み込みまたはデータのアップロードのいずれかの開始準備が完了しているかどうかを確認できますが、両方を同時に確認することはできません。

POST /api/pub/dataIngestion/v1/dataSets/{dataSet}/readyForIngestion

入力: DataIngestionCycle が含むもの

- 更新が予定された、既存の完全に設定済みのデータ ソースに基づく SourceTableDefinitions のいずれか。任意の識別子を指定できます。他のプロパティはオプションであり、無視されます。
- あるいは、データ読み込みの開始を示す論理フラグです。ソース テーブル定義の一覧の指定と、論理フラグを true に設定することは、同時にはサポートされていません。データ読み込みの後にアップロードを実行する必要がある場合は、これらを 2 つのデータ インジェスト サイクルとして別々に実行する必要があります。

出力: IngestionReadyState データの読み込みの準備確認では、すべての既存の検証問題を必ずしも考慮するわけではありません。ユーザー インターフェイスに検証問題が表示されていても、準備確認で確認の成功が報告される場合があります。

1.4.8 新しいインジェスト サイクルの作成

新しいインジェスト サイクルを作成します。

インジェスト サイクルはデータ読み込みまたはデータのアップロードのいずれかの開始に対応して作成されますが、両方に同時に対応して作成されることはありません。

POST /api/pub/dataIngestion/v1/dataSets/{dataSet}/ingestionCycles

入力: DataIngestionCycle が含むもの

- 更新が予定された、既存の完全に設定済みのデータ ソースに基づく SourceTableDefinitions のいずれか。任意の識別子を指定できます。他のプロパティはオプションであり、無視されます。
- あるいは、データ読み込みの開始を示す論理フラグです。ソース テーブル定義の一覧の指定と、論理フラグを true に設定することは、同時にはサポートされていません。データ読み込みの後にアップロードを実行する必要がある場合は、これらを 2 つのデータ インジェスト サイクルとして別々に実行する必要があります。

応答ボディで入力を指定する必要があります。データ読み込みの場合、以下のように応答ボディで "dataLoadTriggered" を true に設定する必要があります。

```
{
  "dataLoadTriggered": true
}
```

出力: 新しい DataIngestionCycle

1.4.9 インジェスト サイクルのキャンセル

既存のインジェスト サイクル (インジェスト サイクル キーによって識別されます) を削除します。

PUT /api/pub/dataIngestion/v1/dataSets/{dataSet}/ingestionCycles/{ingestionCycle}/canceled

出力: キャンセルされた DataIngestionCycle

1.4.10 データのアップロード

指定されているソース テーブルに対応する ARIS Process Mining へデータをアップロードして、ARIS Process Mining の列の順序に基づいて列を自動的に並び替えます。

データの構造 (適切なデータ型と形式を持つ列の数と順序) が適切である必要があります。

POST /api/pub/dataIngestion/v1/dataSets/{dataSet}/sourceTables/{sourceTable}/data

入力:

- パス パラメーターとしてのソース テーブル識別子。識別子はキーまたは完全修飾名のいずれかになります。
- 本体としてのオブジェクトの一覧。新しいソース データ エントリを表します。
  - 列の順序は、ソース テーブルの作成時に指定された順序に対応し、sourceTableDefinitions についての GET 操作によって返されます。
  - タイムスタンプ データは、対応するソース テーブル列の日付と時刻の形式に合わせた文字列のみで渡すことができます。
  - 大量のデータ セットは、複数回のリクエストを利用してアップロードできます。各リクエストのデータは、サーバー側に一時的な形式で保存されます。

出力: エラーなしでデータが受信された場合の成功結果。

1.4.11 ソース テーブルのドロップ

指定されたソース テーブル（定義と内容）をドロップします。

DELETE /api/pub/dataIngestion/v1/dataSets/{dataSet}/sourceTables/{sourceTable}

入力: パス パラメーターとしてのソース テーブル識別子。識別子はキーまたは完全修飾名のいずれかになります。

出力: エラーなしで削除が実行された場合の成功結果。

1.4.12 データ アップロード サイクルのコミット

データ アップロードが完了し、ARIS Process Mining でのインジェストが開始することを ARIS Process Mining に通知します。

ARIS Process Mining でのアップロードがエラーなしで完了すると、インジェスト サイクルのステータスが COMLETED_SUCCESSFULLY に更新されます。これは、データを読み込む新しいインジェスト サイクルを開始するための前提条件です。

PUT /api/pub/dataIngestion/v1/dataSets/{dataSet}/ingestionCycles/{ingestionCycle}/dataComplete

出力: DataIngestionCycle の実行

1.4.13 インジェスト サイクルの抽出

データ セットに対応するすべての既存インジェスト サイクルを抽出します。

GET /api/pub/dataIngestion/v1/dataSets/{dataSet}/ingestionCycles

出力: [DataIngestionCycle] オブジェクトの一覧

エンドポイントは ARIS Process Mining バージョン 10.18 から使用できます。

1.4.14 インジェスト サイクル状態を返す

指定されているインジェスト サイクルの状態を抽出します。

GET /api/pub/dataIngestion/v1/dataSets/{dataSet}/ingestionCycles/{ingestionCycle}/state

出力: 状態の値は、対応する実行ログのエントリに基づきます。

1.5 データ転送オブジェクト（DTO）

データ インジェスト API に以下のデータ転送オブジェクト（DTO）を使用できます。

1.5.1 SourceTableDefinition

入力として

一覧形式のみで、ここで提示されるようにスタンドアロン、またはデータ インジェスト サイクル（DataIngestionCycle）の一部のいずれか（下記参照）。プロパティは、テーブルを作成する必要があるのか、または置き換える必要があるのかによって、必須の場合とオプションの場合があります。

```
[
	{
		"key": "prq_some_namespace_e",
		"name": "example_table_o",
		"namespace": "some_namespace",
		"fullyQualifiedName": "some_namespace.example_table_o",
		"persistenceMode": "OVERWRITE|APPEND",
		"mergeKey": ["PROCESSOR_GROUP", "PROCESSOR"],
		"columns": [
			{
				"dataType": "DOUBLE",
				"name": "CATEGORY"
			},
			{
				"dataType": "STRING",
				"name": "CATEGORY_NAME"
			},
			{
				"dataType": "FORMATTED_TIMESTAMP",
				"name": "CREATED",
				"format": "yyyy/MM/dd HH:mm:ss"
			},
			{
				"dataType": "STRING",
				"name": "PROCESSOR"
			},
			{
				"dataType": "STRING",
				"name": "PROCESSOR_GROUP"
			}
		]
	}
]
```

出力として

一覧形式のみで、ここで提示されるようにスタンドアロン、またはデータ インジェスト サイクル（DataIngestionCycle）の一部のいずれか（下記参照）。

```
[
      {
            "key": "prq_some_namespace_e",
            "name": "example_table_o",
            "namespace": "some_namespace",
            "fullyQualifiedName": "some_namespace.example_table_o",
            "persistenceMode": "OVERWRITE",
            "mergeKey": ["PROCESSOR_GROUP", "PROCESSOR"],
            "columns": [
                  {
                        "dataType": "DOUBLE",
                        "name": "CATEGORY"
                  },
                  {
                        "dataType": "STRING",
                        "name": "CATEGORY_NAME"
                  },
                  {
                        "dataType": "FORMATTED_TIMESTAMP",
                        "name": "CREATED",
                        "format": "yyyy/MM/dd HH:mm:ss"
                  },
                  {
                        "dataType": "STRING",
                        "name": "PROCESSOR"
                  },
                  {
                        "dataType": "STRING",
                        "name": "PROCESSOR_GROUP"
                  }
            ]
      }
]
```

- 鍵はサーバー上で生成されます。
- 完全修飾名（fullyQualifiedName）は、名前と名前空間で構成され、「.」で区切られます。
- 永続モード（persistenceMode）は、OVERWRITE または APPEND のいずれかが可能です。詳細は、「永続モード」『29ページ 』の章を参照してください。
- 列の型は、DOUBLE、LONG、STRING、FORMATTED_TIMESTAMP が可能です。
- マージ キー（mergeKey）はオプションです。マージ キーは、新しいデータを既存データとマージするときのみ必要です。

注意

ソース テーブル定義が作成または更新される際に、[タイムスタンプ] 型の [_ARIS_ lastChanged] 列が必ずアップロードされたテーブルに自動的に追加されます。_ARIS_ プレフィックスは、内部使用のために予約されています。名前が、_ARIS_ で始まる列を指定しないでください。そのような列名を含むテーブルを作成するリクエストはエラーになります。

ソース テーブル定義が API から抽出される際に、名前が _ARIS_ で始まる列は除外されます。API を使用してソース テーブル定義を更新していた場合（置換した場合とは対照的に）、その後の GET は、その更新によって設定された列のみを返し、[removed] 列を除外します。ただし、それらの列と、それらの列の既存データは、当然ながらそのまま存在します。ARIS Process Mining 内でもそのまま処理されます。これは、API クライアントが常に、最新の形の（最新のスキーマを使用する）テーブルを確実に受け取り、それにとって既知の列のみが含まれるようにするためです。

1.5.2 DataIngestionReadyState

準備確認後の出力としてのみ使用されます。準備が完了していない場合は ready プロパティに false が設定され、オブジェクトに原因がコードとメッセージとともに含まれます。

```
{
  "ready": false,
  "cause": {
    "code": "INR1001",
    "message": "The data set is currently being processed"
  }
}
```

原因は、データ セットの準備が完了していない理由を示すコードとメッセージで構成されます。コードは 4 桁長で、必ず「Ingestion - Not Ready」(インジェスト - 準備未完了）を示す接頭辞 INR が付きます。以下のテーブルは、具体的なコードと意味の一覧です。

コード	意味
INR1000	未定義。 想定外の状況で使用されました。
INR1001	データ セットが処理中です。 参照されたデータ セットが別のプロセス（データ インジェスト サイクル、手動でのデータの読み込みなど）によってロックされています。 ユーザは、ロックが解除されるまで待機し、リクエストを繰り返す必要があります。

コード	意味
INR1002	Living process の割当を超えました。 データ インジェスト API の使用は、「Living Process」ライセンスがあるデータ セットに限定されます。 割り当てられている「Living Process」ライセンスは、対応するデータ セット内のプロセス インスタンスの数に上限（プロセス割当）を定義します。 コード INR 1002 は、割り当てられている「Living Process」ライセンスによって定義されたプロセス割当が既に超過していて、これ以上のデータを取り込むことができないことを示します。ユーザーは、プロセス インスタンスを削除してデータ セット内のプロセス インスタンスの数を減らすか、プロセス割当がさらに大きな「Living Process」ライセンスを代わりに割り当てることができます。
INR1003	予期しないソース テーブル タイプ。 1 つ以上の参照されたソース テーブルに、予期しないタイプ（CSV のアップロードで作成されたタイプなど）があることを示します。 データ インジェスト API のコンテキストでテーブルを使用するには、API で作成されているか、ソース テーブルの置換機能で更新されているかのいずれかが必要です。
INR1004	データが読み込まれていません。 データ モデリング セクションで参照されているテーブルに、新しい保留中のデータがないことを示します。そのため、データの読み込みは不要です。

1.5.3 DataIngestionCycle

入力として

データをアップロードするケース

```
{
    "dataUploadTargets": [
        {
            "fullyQualifiedName": "some_namespace.example_table_a"
        }
    ]
}
```

データを読み込むケース

```
{
    "dataLoadTriggered": true
}
```

出力として

一覧形式でのいずれか

```
[
    {
        "key": "api_2",
        "dataUploadTargets": [
            {
                "key": "prq_some_namespac_38",
                "name": "example_table_a",
                "namespace": "some_namespace",
                "fullyQualifiedName": "some_namespace.example_table_a",
                "persistenceMode": "APPEND",
                "columns": [
                    {
                        "dataType": "DOUBLE",
                        "name": "CATEGORY"
                    },
                    {
                        "dataType": "STRING",
                        "name": "CATEGORY_NAME"
                    },
                    {
                        "dataType": "FORMATTED_TIMESTAMP",
                        "name": "CREATED",
                        "format": "yyyy/MM/dd HH:mm:ss"
                    },
                    {
                        "dataType": "STRING",
                        "name": "PROCESSOR"
                    },
                    {
                        "dataType": "STRING",
                        "name": "PROCESSOR_GROUP"
                    }
                ]
            }
        ],
        "dataLoadTriggered": false,
        "state": {
            "value": "INGESTING_DATA"
        }
    },
    {
        "key": "api_1",
        "dataLoadTriggered": true,
        "state": {
            "value": "COMPLETED_SUCCESSFULLY"
        }
    }
]
```

あるいは、例を挙げると、作成、更新、キャンセル後のスタンドアロン。

データをアップロードするケース


```
{
    "key": "api_1",
    "dataUploadTargets": [
        {
            "key": "prq_some_namespac_38",
            "name": "example_table_a",
            "namespace": "some_namespace",
            "fullyQualifiedName": "some_namespace.example_table_a",
            "persistenceMode": "APPEND",
            "columns": [
                {
                    "dataType": "DOUBLE",
                    "name": "CATEGORY"
                },
                {
                    "dataType": "STRING",
                    "name": "CATEGORY_NAME"
                },
                {
                    "dataType": "FORMATTED_TIMESTAMP",
                    "name": "CREATED",
                    "format": "yyyy/MM/dd HH:mm:ss"
                },
                {
                    "dataType": "STRING",
                    "name": "PROCESSOR"
                },
                {
                    "dataType": "STRING",
                    "name": "PROCESSOR_GROUP"
                }
            ]
        }
    ],
    "dataLoadTriggered": false,
    "state": {
        "value": "INGESTING_DATA"
    }
}
```


データを読み込むケース


```
{
    "key": "api_1",
    "dataLoadTriggered": true,
    "state": {
        "value": "INGESTING_DATA"
    }
}
```

1.5.4 DataIngestionCycleState

出力としてのみ使用されます。スタンドアロンまたはデータ インジェスト サイクルの一部 (DataIngestionCycle) (上記参照)。可能な状態は次のとおりです。ACCEPTING_DATA、INGESTING_DATA、COMPLETED_SUCCESSFULLY、CANCELED、FAILED。

FAILED の場合は、原因を返します。原因は、何が起きたのかを正確に示すコードとメッセージで構成されます。コードは 4 桁長で、必ず「Ingestion - Error」(インジェスト - エラー) を示す接頭辞 IER が付きます。

```
{
  "value": "FAILED",
  "cause": {
    "code": "IER1000",
    "message": "An unexpected error occurred"
  }
}
```

TABLEDATA

データのアップロードの入力としてのみ使用されます。値は、目的のソース テーブルのスキーマに適合する必要があります。Null は有効な値です。

```
[
    [1,"A","2021/05/10 12:13:14",1.1,"Distribution Center
Team","Distribution"],
    [2,"B","2021/06/11 15:16:17",2.2,"Distribution Center
Team","Distribution"],
    [3,"C","2021/07/12 18:19:20",3.3,null,"Sales"],
    [4,"D","2021/08/13 21:22:23",4.4,"Dealer Sales","Sales"]
]
```

STRINGCOLUMN

ソース テーブル定義 (SourceTableDefinition) の一部としてのみ使用されます (上記参照)。

LONGCOLUMN

ソース テーブル定義 (SourceTableDefinition) の一部としてのみ使用されます (上記参照)。

DOUBLECOLUMN

ソース テーブル定義 (SourceTableDefinition) の一部としてのみ使用されます (上記参照)。

FORMATTEDTIMESTAMPCOLUMN

ソース テーブル定義 (SourceTableDefinition) の一部としてのみ使用されます (上記参照)。

DEFAULTRESULT

スタンドアロン出力としてのみ使用されます。実行された操作に専用の結果オブジェクト自体がない場合 (ソース テーブルの削除またはソース データのアップロード)、またはサーバーでエラーが発生した場合 (任意の操作) のいずれかです。このオブジェクトの成功した場合のプロパティに、状況に応じて true または false のいずれかが設定されます。

false の場合は、オブジェクトに原因とメッセージも含まれます。

```
{
  "successful": false,
  "cause": {
    "message": "An unexpected error occurred"
  }
}
```

APIVERSION

API バージョンの確認後に、出力としてのみ使用されます。

```
{
  "apiVersion":「3.2」
}
```

1.5.5 認証応答

ARIS Cloud に対する認証リクエストの応答は、テナント、URL、アクセス トークンを含む JSON オブジェクトです。

{

"tenant": "<project_room>",

"token": "<access_token>,

"url": "<any_URL>"

}

例

```
{
"tenant": "myProjectRoom",
"token":"...eyJpYXQiOjE2NjE5MzY3NzgsImp0aSI6IjBqLWg2TkZqc3RLb0pTZ1U1dXJUY
mRXcUs3NGplRV9EZzRyeXhOeDN5dkxkakJsRFI2Z2NzUEJueGpRTmNHTXU0cFo2R2loazMwQ0
NMOUR4d0lQdiIsInN1YiI6ImR...",
"url": "https://processmining.ariscloud.com"
}
```

ARIS Enterprise Cloud に対する認証リクエストの応答はアプリケーション トークンを含む JSON オブジェクトです。

```
{
'applicationToken': '…'
}
```

1.6 重要情報

1.6.1 永続モード

すべてのソース テーブルには、サーバーでの新しいデータの処理方法を決定する永続モードがあります。

3 つの異なる永続モードがあります。

OVERWRITE

これは、以前のバージョンからの標準的な振る舞いに従うデフォルト設定です。このモードが設定されていると、新しくアップロードされたデータで既存のデータを上書きします。上書きされたデータは失われ、復元できません。必要な場合は、上書きされたデータを再びアップロードしなければなりません。

APPEND (マージ キーなし)

このモードが設定されていると、新しくアップロードしたデータはサーバーに残っているテーブル データを上書きするのではなく、既存データに追加されます。新しいデータ行が、受信した順序で最後に追加されます。これは、既存のソース テーブルのサイズが大きくなる原因になります。この設定で古いデータの持続が 2 回目になると (新しい行であるかのように)、エントリが重複することになります。これは、分析結果の正確性に影響を与える場合があります。

現在のところ、このモードをソース テーブルに選択する唯一の方法は、データ インジェスト API を使用して、新しいテーブルを作成するか、既存のテーブルを置き換えるしかありません。

APPEND (マージ キーあり)

このモードが設定されると、アップロードされたデータはサーバー上にある既に永続的なテーブル データとマージされます。マージ モードでは、新しいデータはテーブルの追加行に追加されます。テーブルの既存行は、アップロードされたテーブルの対応する行の方が新しい場合のみ、個別に上書きされます。データ アップロードが終了した時点で、ソース テーブルにはすべてのデータが含まれます。

インジェスト API はマージ キーを使用して、ソース テーブルで既存データと新しいデータをマージします。ソース テーブル定義にマージ キーを設定する『21ページ 』と、マージ モードを使用できます。また、ソース テーブル定義に APPEND するには永続モード (persistenceMode) を設定する必要もあります。

既存テーブルに対してマージ キーを設定するには、新しいマージ キーとともにソース テーブル定義をサーバーに送信する必要があります。そのために、[ソース テーブルの作成または置換]『16ページ 』終了ポイントを使用できます。マージ キーは、ソース テーブルが作成または置換される際『16ページ 』に、自動的にソース テーブルに追加されます。

注意

OVERWRITE 永続モードを持つソース テーブルは、ARIS Process Mining の標準テーブルに対応し、指定されたマージ キーを使用する APPEND 整合性モードを持つソース テーブルは、それに応じた増分テーブルに対応します。

1.6.2 制限

リクエスト サイズ

データを作成または更新するリクエストの最大許容サイズは 100 MB に制限されています。この最大値を超えると、そのリクエストは拒否されます。さらに大きなデータを作成または更新する場合は、データを複数のリクエストに分割します。

ソース テーブル

ソース テーブルの合計数

データ インジェスト API を使用して作成できるソース テーブルの最大数は 100 です。API を使用して新しいテーブルが作成されるたびに、最大数を超えていないか確認されます。この数を超えると、該当するリクエストは拒否されます。すべての既存テーブルは、その起点（API、抽出、手動でのファイルのアップロード）が何であれ、最大許容数に向けてカウントされます。この制限はソース テーブルの置換には影響を及ぼしません。

1 リクエストあたりのソース テーブルの数

1 回のリクエストで作成できるソース テーブルの最大数は 50 です。この数を超えると、該当するリクエストは拒否されます。

列の数

データ インジェスト API を使用してソース テーブルに作成できる列の最大数は 500 です。この数を超えると、該当するリクエストは拒否されます。この制限はソース テーブルの作成と置換の両方には影響を及ぼしません。

タスクの合計数

同時に維持できるタスク（インジェスト サイクルを含む）の最大数は 350 です。API を使用して新しいテーブルが作成されるたびに、最大数を超えていないか確認されます。この数を超えると、該当するリクエストは拒否されます。依然として維持されているすべての既存タスクは、そのタイプ（インジェスト サイクル、抽出、手動でのファイルのアップロード、データ読み込み、再計算、プロセス データの削除）や起点（API、自動化、手動実行）が何であれ、最大許容数に向けてカウントされます。維持されているタスクは、30 分間隔で自動的にクリーンアップされます。クリーンアップ ルーチンは最新の 250 エントリを除くすべての完了済みタスクを削除します。

アップロード

データ アップロード ターゲット数

データ インジェスト API の 1 回のアップロード サイクルで参照できるソース テーブル (データ アップロード ターゲット) の最大数は 100 です。この最大数を超えると、該当するリクエストは拒否されます。アップロードが必要なソース テーブルがさらにある場合は、複数のアップロード サイクルに分割する必要があります。

保留中のデータ パッケージ数

データ インジェスト API を使用してデータをアップロードする場合、1 つのテーブルについて保留中のアップロード データ パッケージの最大許容数は 50 です。この数を超えると、該当するリクエストは拒否されます。目的のテーブルにさらに多くのデータをアップロードする場合は含まれるインジェスト サイクルを完了に設定して、サーバー側の永続性を開始します。永続性 (およびインジェスト サイクル) が完了した後に、残りのデータをアップロードするための新しいサイクルが作成できるようになります。2 番目のアップロード サイクルは、目的のテーブルの永続モードが APPEND に設定されている場合のみ直ちに開始されるので注意してください。モードに OVERWRITE が設定されている場合は、まず、データを読み込む必要があります (読み込みサイクル)。データを読み込んだ後のみ、残りのデータを安全に読み込むことができます。

1.7 ARIS Process Mining 用 webMethods.io コネクタ

ARIS Process Mining 用 webMethods.io コネクタは、任意のデータ ソースから ARIS Process Mining にデータを転送するためにデータ インジェスト API を使用します。ARIS Process Mining 用 webMethods.io コネクタを使用すれば、データの作成、作成されたテーブルへのデータのアップロード、ARIS Process Mining でのデータ読み込み操作のトリガーなどが可能になります。

あらかじめ定義された操作によって、最も一般的な REST リソースと操作を直接使用したり、REST 操作をカスタマイズする際の複雑さを軽減したりできます。

ARIS Process Mining 用 webMethods.io コネクタの使用については、webMethods.io 文書を参照してください。

以下の一覧には、ARIS Process Mining 用 webMethods.io コネクタによって提供されるすべての事前定義済み操作が含まれ、操作に参照されるデータ インジェスト エンドポイント『14ページ 』が記載されています。

操作	エンドポイントへの参照
API バージョンの抽出	API バージョンの抽出『16ページ 』
ソース テーブル定義の抽出	ソース テーブル定義の抽出『16ページ 』
ソース テーブルの作成	ソース テーブルの作成または置換『16ページ 』
ソース テーブルの置換	ソース テーブルの作成または置換『16ページ 』

操作	エンドポイントへの参照
ソース テーブル定義の更新	ソース テーブル定義の更新『17ページ 』
データ アップロードの準備確認	データ セットがインジェストの準備を完了しているかどうかを確認する『18ページ 』
データ読み込みの準備確認	データ セットがインジェストの準備を完了しているかどうかを確認する『18ページ 』
データ アップロード サイクルの作成	新しいインジェスト サイクルの作成『18ページ 』
データ読み込みの開始	新しいインジェスト サイクルの作成『18ページ 』
サイクル状態の抽出	インジェスト サイクルの抽出『20ページ 』
サイクルのキャンセル	インジェスト サイクルのキャンセル『19ページ 』
データ アップロード サイクルのコミット	データ アップロード サイクルのコミット『20ページ 』
ソース テーブルのドロップ	ソース テーブルのドロップ『20ページ 』
サイクルの抽出	インジェスト サイクルの抽出『20ページ 』
データのアップロード	ソース テーブル定義の抽出『16ページ 』 データのアップロード『19ページ 』

2 サポートおよび法的情報

このセクションでは、製品サポートおよび法的観点に関する一般情報を説明します。

2.1 ドキュメンテーションのスコープ

提供されている情報では、印刷が行われた時点における設定および機能について説明しています。ドキュメンテーションとソフトウェアの生産サイクルが異なるため、設定や機能に関する説明が、実際の設定や機能と異なることがあります。相違に関する情報は製品に付属しているリリース ノートに記載されています。リリース ノートをお読みになり、記載されている情報を考慮して製品をインストール、設定、および使用してください。

Software GmbH によって提供されるコンサルティング サービスを利用せずにシステムの技術的機能と業務機能をインストールする場合は、インストールするシステム、その目的、対象システム、さまざまな依存性などに関して広範な知識が必要です。プラットフォームの数が多く、ハードウェアとソフトウェアの設定が相互に依存するので、特定のインストール シナリオしか説明できません。すべての設定と依存性を記述することはできません。

各種の技術を組み合わせる場合は、製造元の指示（特にインターネット ページに公開されたリリースに関するお知らせ）に従ってください。承認されているサードパーティ システムが正しく機能すること、および正しくインストールされることの保証はいたしかねます。また、サードパーティ システムはサポートしていません。必ず、該当の製造元のインストール マニュアルに記載されている手順に従ってください。問題がある場合は、製造元にお問い合わせください。

サードパーティ システムのインストールにサポートが必要な場合は、最寄りの Software GmbH の販売部門にお問い合わせください。このような製造元またはお客様固有の変更は、Software GmbH の標準ソフトウェア保守契約の対象ではありません。このような変更は、それを特別に要請し、同意した場合にのみ実行できます。

2.2 データ保護

Software GmbH の製品は、個人データの処理に関して EU 一般データ保護規則 (GDPR) に従う機能を備えています。

該当する場合、各管理文書において適切な手順が文書化されています。

2.3 サポート

ユーザーが実行できない特定の機能について質問がある場合は、最寄りの Software GmbH の営業担当『 https://www.softwareag.com/corporate/company/global/offices/default.htmlsee 』までお問い合わせください。詳細情報とサポートについては、弊社の Web サイトをご利用ください。

サポート契約が有効な場合は、グローバル サポート ARIS (+800 ARISHELP) までお問い合わせください。ご利用の電話会社でこの番号が使用できない場合は、弊社のグローバル サポートの連絡先ディレクトリを参照してください。

製品の文書に関して問題がある場合は、documentation@softwareag.com『 mailto:documentation@softwareag.comsee 』までメールを送信することもできます。

ARIS COMMUNITY

- 製品、更新、修正のダウンロード
- 情報、専門家の記事、問題解決、ビデオ、他の ARIS ユーザーとのコミュニケーションが見つかります

アカウントをまだお持ちでない場合は、ARIS Community にてご登録ください。

製品トレーニング

Learning Portal (ラーニング ポータル) に有用な製品トレーニング資料があります。

テック コミュニティ

弊社のテック コミュニティ Web サイトにて、Software GmbH の専門家とコラボレーションできます。以下は、ここでできることです。

- 膨大なナレッジ ベースの閲覧
- ディスカッション フォーラムでの質問し、答えを見つける
- 最新の Software GmbH ニュースと発表の入手
- コミュニティの調査

- 弊社のパブリック GitHub と Docker リポジトリにアクセスして、追加的な Software GmbH リソースを発見する

製品サポート

Software GmbH 製品のサポートは、ライセンスを付与されたお客様に対して弊社の Empower Portal『 https://empower.softwareag.com/see 』を通じて提供されます。このポータルの多くのサービスは、アカウントを必要とします。またお持ちでない場合は、申請できます。アカウントがあれば、以下ができます。

- 製品機能の追加リクエスト
- ナレッジ センターで技術情報とワンポイントを検索する
- 早期の警告と重要な警告のサブスクリプション
- サポート インシデントの開始と更新