ARIS PROCESS MINING

DATA INGESTION API

VERSION 10.0 - SERVICE RELEASE 27 AND HIGHER

OCTOBER 2024

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.

Contents

1 Ingest data using a public API

ARIS Process Mining supports a public data ingestion API. You create and send HTTP requests to use the API. The API allows you to transfer data from any data source to ARIS Process Mining. The data transferred to ARIS Process Mining is in a logical tabular structure and must conform to JSON format.

You can use an appropriate API client to create HTTP requests and you need the appropriate API programming skills.

The Data transfer objects (DTOs) (page 21) chapter lists the DTOs that you can use for transferring data.

The API Methods (page 15) chapter lists the endpoints that you can use for your HTTP requests.

PREPARATIONS IN ARIS PROCESS MINING

To be able to transfer data to ARIS Process Mining using an API, you must perform the following steps:

- Create a system integration for the data ingestion API (page 2).
- Create a data set to store the transferred data.
- Create a connection for the data ingestion API. (page 3)

USE THE DATA INGESTION API TO TRANSFER DATA

The following steps are a best practice for transferring data using an API.

- Authenticate your API client (page 5)
- Retrieve source table definitions (page 9)
- Create or replace source tables (page 9)
- Check if the data set is ready for data upload (page 10)
- Create a data upload cycle (page 11)
- Upload data (page 12)
- Commit data upload cycle (page 12)
- Retrieve cycle state (page 13)
- Check if the data set is ready for data load (page 13)
- Start the data load (page 13)
- Retrieve cycle state (page 14)

If required, you can also perform the following steps.

- Drop source table (page 14)
- Retrieve ingestion cycles (page 14)

- Cancel ingestion cycle (page 14)

1.1 Notes

Note that new JSON fields and enumeration values may be added to Ingestion API inputs and outputs by Software GmbH as part of minor revisions to the API.

1.2 Preparations in ARIS Process Mining

1.2.1 Create a system integration for the data ingestion API

To use the data ingestion API (page 1), you must create a corresponding system integration.

ARIS Process Mining supports the OAuth2 Flows with **Client credentials** and **Authorization code** grant types as authentication methods.

The authentication methods with client credentials is outside of a context of a user and is recommended for machine-to-machine communication.

Prerequisite

You have installed the ARIS Process Mining Enterprise license.

Procedure

1. Click the ⁝⁝⁝ **Navigation menu** icon > **Administration** in the program header.
2. Click **System integration** in the **Administration** panel.
3. Click **Add system integration** > **Data ingestion (API)**. The corresponding dialog opens.
4. Enter a name, for example, Data ingestion, and an optional description.
5. Select an authentication method in the **Grant type (OAuth)** drop-down menu.

   Client credentials as authentication method is recommended. It is intended for machine-to-machine communication and is outside of the context of an actual log-in user.

   If you select the **Authorization code** grant type, specify the **Authorization callback URL** that is used for authentication.

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

   Replace hostname <region.ariscloud> with the hostname of the ARIS Process Mining installation and the <project_room> with the ARIS Process Mining project room you want to login to.

You can read the hostname of the URL (for example, processmining.ariscloud.com) in the browser address bar if you are logged in.

Examples

Authorization callback URL for the ARIS cloud

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

Authorization callback URL for the ARIS Enterprise cloud

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

6. Click **Add**. The **Data ingestion access data** dialog opens. The dialog provides the client ID, secret key, and project room name.

   If you have selected the **Authorization code** grant type the well-known URL is additionally shown.

7. You can save the provided authentication data, for example, using a text editor.

   Click **Copy to clipboard** and save the data.

8. Click **Done**.

The system integration is created and listed with the name you specified.

Note that the system integration of the data ingestion API remains in the **Pending** state by default. However, you can use the system integration properly.

Tip

The access data (except for the endpoints) is saved in the system integration you created. You can display the source system access data to access the client credentials key.

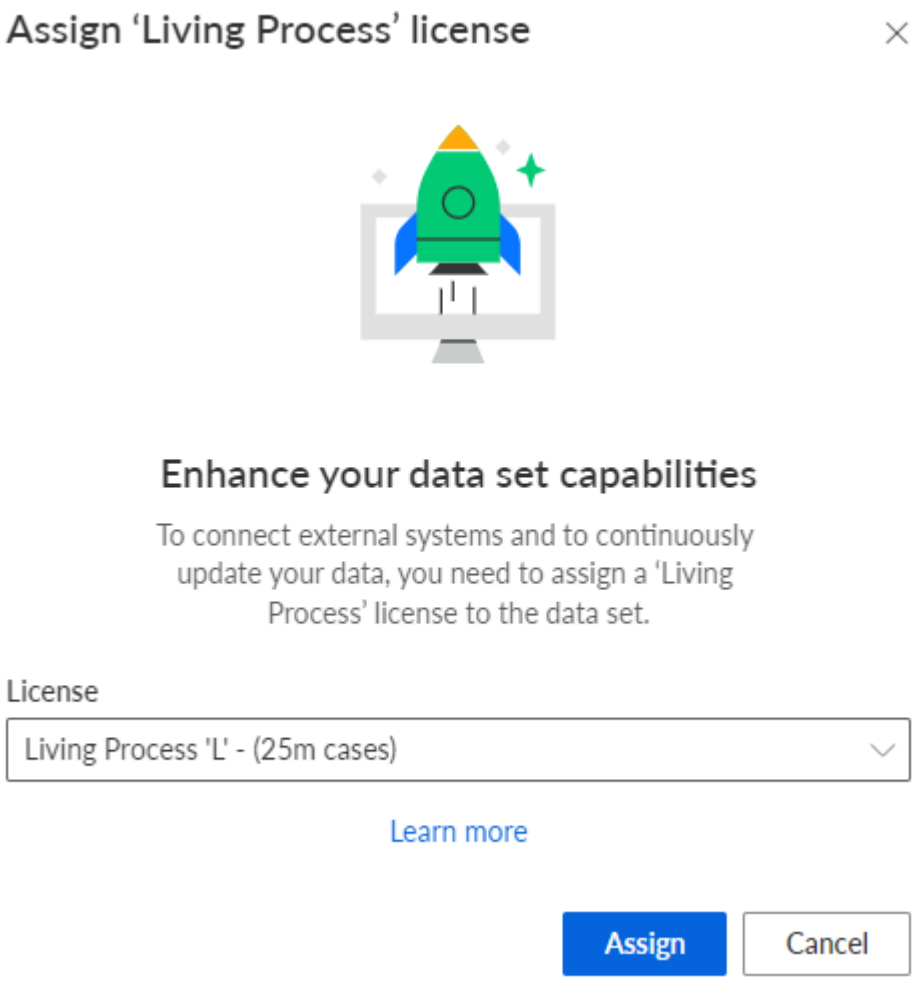
1.2.2 Create a connection in the data set

Before you can transfer data to ARIS Process Mining using the data ingestion API, you must create a corresponding connection for the data set where the transferred data is stored. You create a connection to the API client using the system integration you created (page 2).

Procedure

1. Click the ⁝⁝⁝ **Navigation menu** icon in the program header.
2. Select ⛁ **Data collection**. The **Data collection** opens and shows the **Data sets** page.
   a. If you have already opened a data set, the recently opened data set is opened. Click **Back** in the **Data set** panel to open the **Data set** page.
   b. Click the data set on the **Data sets** page. The selected data set opens.

3. Open the **Connections** component.
4. Click **Add connection**. If you add a connection to a source system for the first time and you have not assigned a 'Living Process' license to the data set yet, the **Assign 'Living Process' license** dialog opens.
5. Select a license in the drop-down menu. You need the 'Living Process' license to extract and analyze processes. The number of processes you can extract depends on the selected license.



6. Click **Assign**. The **Add connection** dialog opens.
7. Configure the connection.
   a. Enter a unique name for the connection to the source system, for example, Data ingestion.
   b. Select the system integration created for the data ingestion API.
   c. Click **Add**.

You have created a connection for the API. The created connection is displayed on the **Connections** page with the settings you specified.

1.3 Use the data ingestion API

The following steps are a best practice for transferring data using the data ingestion API.

The operations described below are available as predefined operations when using the webMethods.io connector for ARIS Process Mining (page 31).

1.3.1 Authenticate your API client

You must perform an HTTP authentication request to authenticate your client against ARIS Process Mining. Depending on the specified authentication method (page 2), you can use client credentials or an authorization code.

You can find the required data in the system integration created for the data ingestion API (page 2).

AUTHENTICATION AGAINST ARIS CLOUD USING CLIENT CREDENTIALS

Note that we firmly recommend the authentication using client credentials.

If you login to your project room using the URL **mc.ariscloud.com**, you are using the ARIS cloud.

Send an HTTP Post request to the ARIS cloud endpoint and path **/api/applications/login** (for example, https://mc.ariscloud.com/api/applications/login) with the following properties:

- Content type: application/x-www-form-urlencoded
- Request body with the values from the corresponding system integration (page 2):

  clientId: client ID

  clientSecret: client secret

  tenant: project room name

The response is a JSON object that consists of the tenant, a URL, and an access token.

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

Note

Use the value for the URL as the hostname for all subsequent calls to the REST endpoints.

Ensure that the generated bearer token is sent with the appropriate header for every subsequent request. To do this, add this HTTP-request header to each request as follows:

```
Authorization: Bearer <token from response>
```

Please note the blank space after the term "Bearer".

AUTHENTICATION AGAINST ARIS ENTERPRISE CLOUD WITH ARIS USER MANAGEMENT USING CLIENT CREDENTIALS

Note that we firmly recommend the authentication using client credentials.

Send an HTTP Post request to ARIS User Management using the path **/umc/api/oauth/apptoken** (for example, https://my_company_name.ariscloud.com/umc/api/oauth/apptoken) with the following properties:

- Content type: application/x-www-form-urlencoded
- Request body with the values from the corresponding system integration (page 2):

  client_id: client ID

  client_secret: client secret

  tenant: project room name

  grant_type: client_credentials

The response is a JSON object that consists of an application token:

```
{
"applicationToken": "…"
}
```

Ensure that the generated bearer token is sent with the appropriate header for every subsequent request. To do this, add this HTTP-request header to each request as follows:

```
Authorization: Bearer <token from response>
```

Please note the blank space after the term "Bearer".

AUTHENTICATION USING AUTHORIZATION CODE

Note that your client application must support OAuth 2.0 with **Authorization code** grant type.

Configure the client application to use:

- **Callback URL**

  The callback URL to which you will be redirected to authenticate against your project room in ARIS Process Mining:

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

You can read the hostname of the URL (for example, processmining.ariscloud.com) in the browser address bar if you are logged in.

Examples

If your project room is in the ARIS cloud, the callback URL could be as follows.

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

If your project room is in the ARIS Enterprise cloud, the URL could be as follows.

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

- **Client ID and client secret**

  You noted them when you created the system integration in ARIS Process Mining, or you can retrieve them from the list in the **System Integration** module in ARIS Process Mining Administration when you view the system access data for that system integration. Client ID and secret must be sent in the body, not as a Basic OAuth header.

- **Authorization, token, and refresh endpoints** can be retrieved by calling the corresponding well-known URL in your browser. You can retrieve the well-known URL from the list in the **System Integration** module in the ARIS Process Mining Administration when you view the system access data for that system integration. The URL will return a JSON object with authorization_endpoint, token_endpoint, refresh_endpoint, and userinfo_endpoint.

  Example

```
{
"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>"}
}
```

After the client application sends an authentication request using these properties, the server responds with a JSON object that consists of the tenant, a URL, and an access token.

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

Note

Use the value for the URL as the hostname for all subsequent calls to the REST endpoints.

Ensure that the generated bearer token is sent with the appropriate header for every subsequent request. To do this, add this HTTP-request header to each request as follows:

```
Authorization: Bearer <token from response>
```

Please note the blank space after the term "Bearer".

Additionally, for the authentication type **Authorization Code** a CSRF token must be sent with each request.

You can acquire a CSRF token after a successful authentication by sending an HTTP POST request to ARIS User Management at the path **/umc/api/v2/tokens/csrftoken**.

The result is a string of alphanumeric characters based on your current user session, for example, oehltw0drUujSdWMD5TJEsXSLklwk1xKYh1LHaZ16g7. You must send this token with the csrftoken header for each subsequent request.

1.3.1.1 Authentication via URL parameters is deprecated

Please note that from Service Release 28 (SR 28), the passing of authentication credentials via URL query parameters is no longer supported. The credentials must be passed in the request body. (page 5)

1.3.1.2 Note the technical key of a data set

All subsequent sections present different requests that are performed in the context of a specific data set. For almost all requests, you must use the technical key of the data set. (page 15)

A request contains the technical key as follows:

/dataSets/<data Set>

The **<dataSet>** parameter refers to the technical key of the data set and uses that value at runtime. You can take the value from the URL in the address bar of the browser when opening the corresponding data set.

The URL has the following format:

https://<hostname>/#<project_room>/dataCollection/y.dataset.<key>

The <key> parameter is based on the selected display name of the data set and should be readable. Use this key in all your API requests to this specific data set.

Example

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

1.3.2 Retrieve source table definitions

To retrieve source table definitions, perform the following HTTP request.

GET "https://<hostname>/mining/api/pub/dataIngestion/v1/dataSets/<data set>/sourceTableDefinitions[?fullyQualifiedNames=default.table_a[,default.table_b]]"

If no fullyQualifiedNames are specified, the structure of all available source tables is returned.

1.3.3 Create or replace source tables

To create or replace source tables, perform the following HTTP requests.

CREATE SOURCE TABLES

POST "https://<hostname>/mining/api/pub/dataIngestion/v1/dataSets/<data set>/sourceTables"

You must send a request body to the server in the following form (here with sample data):

```
[
  {
      "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": [
       ...
      ]
  }
]
```

REPLACE SOURCE TABLES

POST "https://<hostname>/mining/api/pub/dataIngestion/v1/dataSets/<data set>/sourceTables?forceReplace=true"

You must send a request body to the server in the following form (here with sample data):

```
[
  {
      "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"
  }
]
```

If no persistence mode (page 29) (persistenceMode) is set, the table is created or replaced with persistenceMode = OVERWRITE.

If you are using the WebMethods connector for ARIS Process Mining, the forceReplace parameter is implicitly set.

1.3.4 Check if the data set is ready for data upload

The data set must be ready to upload the data. To check the state of the data set, perform the following HTTP request.

POST "https://<hostname>/mining/api/pub/dataIngestion/v1/dataSets/<data set>/readyForIngestion"

You must send a request body to the server in the following form (here with sample data):

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

If the data set is ready, you receive a positive response. Otherwise, the response is negative and contains the corresponding reason.

Example

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

1.3.5 Create a data upload cycle

If the data set is ready, you can create a data ingestion cycle for the data upload with the following HTTP request.

POST "https://<hostname>/mining/api/pub/dataIngestion/v1/dataSets/<data set>/ingestionCycles"

You must send a request body to the server in the following form (here with sample data):

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

The response for the call above returns the fully-formed data ingestion cycle, containing a technical key, the referenced data upload targets (tables), and some state information. It's initial state is **ACCEPTING_DATA**. All tables referenced by the cycle are locked for everything but the upcoming data upload.

```
{
    "key": "data_set_1_55",
    "dataUploadTargets": [...],
    "dataLoadTriggered": false,
```

```
    "state": {
        "value": "ACCEPTING_DATA"
    }
}
```

Note down the cycle's technical key and use it for all subsequent requests performed in the context of this cycle, for example, when committing the data upload cycle.

A request contains the technical key as follows:

/ingestionCycles/<ingestion cycle>

1.3.6 Upload data

To upload the data of a source table to the data set, perform the following HTTP request.

POST "https://<hostname>/mining/api/pub/dataIngestion/v1/dataSets/<data set>/sourceTables/<source table>/data"

<source table>: fully qualified name (default.table_a)

You must send a request body to the server in the following form (here with sample data):

```
[
  ["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"]
]
```

Larger amounts of data can be uploaded by means of multiple requests. With each request, the data is stored in temporary form on the server.

1.3.7 Commit data upload cycle

Indicates ARIS Process Mining that all data has been uploaded and can be loaded into the source tables.

PUT "https://<hostname>/mining/api/pub/dataIngestion/v1/dataSets/<data set>/ingestionCycles/<ingestion cycle>/dataComplete"

The ingestion cycle state changes to **INGESTING_DATA**. The uploaded temporary data is now persisted in the source database.

The state of the specified cycle will be set to "COMPLETED_SUCCESSFULLY" when the upload has finished without errors in ARIS Process Mining.

1.3.8 Retrieve cycle state

To read the current cycle state, perform the following HTTP request.

GET "https://<hostname>/mining/api/pub/dataIngestion/v1/dataSets/<data set>/ingestionCycles/<ingestion cycle>/state"

If the data ingestion failed, you receive a response with the corresponding reason:

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

Otherwise, the ingestion status can be **INGESTING_DATA** if the cycle is still running, **COMPLETED_SUCCESSFULLY** if it went through without any problems, or **CANCELED** if it was aborted (for example, using the API).

1.3.9 Check if the data set is ready for data load

The data set must be ready to load the data into the process storage. To check the state of the data set, perform the following HTTP request.

POST "https://<hostname>/mining/api/pub/dataIngestion/v1/dataSets/<data set>/readyForIngestion"

You must send a request body to the server in the following form:

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

If the data set is ready, you receive a positive response. Otherwise, the response is negative and contains the corresponding reason

1.3.10 Start the data load

If the data set is ready for data load, create a data ingestion cycle to start the data load using the following HTTP request.

POST "https://<hostname>/mining/api/pub/dataIngestion/v1/dataSets/<data set>/ingestionCycles"

You must send a request body to the server in the following form:

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

```
}
```

The response for the call above returns the fully-formed data ingestion cycle. The initial state is **INGESTING_DATA**. The corresponding data load starts immediately.

Note that if you use the webMethods connector for ARIS Process Mining, the "dataLoadTriggered" parameter is implicitly set.

1.3.11 Retrieve cycle state

To verify that the data load was successful, perform the following HTTP request.

GET "https://<hostname>/mining/api/pub/dataIngestion/v1/dataSets/<data set>/ingestionCycles/<ingestion cycle>/state"

You receive a response with the current state value (**INGESTING_DATA**, **COMPLETED_SUCCESSFULLY**, or **FAILED**) and optionally a respective reason.

For more details, see chapter Retrieve cycle state (page 13).

1.3.12 Drop source table

Drops the specified source table.

DELETE "https://<hostname>/mining/api/pub/dataIngestion/v1/dataSets/<data set>/sourceTables/<source table>"

1.3.13 Retrieve ingestion cycles

Retrieves all existing ingestion cycles.

GET "https://<hostname>/mining/api/pub/dataIngestion/v1/dataSets/<data set>/ingestionCycles"

1.3.14 Cancel ingestion cycle

Cancels the specified cycle.

PUT "https://<hostname>/mining/api/pub/dataIngestion/v1/dataSets/<data set>/ingestionCycles/<ingestionCycle>/canceled"

1.4 API Methods

You can use the following endpoints provided by the data ingestion API for your HTTP requests.

1.4.1 Path Section: Data Set

All endpoints, except the API version endpoint, are used in the context of a specific data set. All endpoints with a data set context contain the following URL section:

/dataSets/{dataSet}

The **{dataSet}** parameter refers to the technical key of the data set and uses that value at runtime. You can take the value from the URL in the address bar of the browser when opening the corresponding data set.

The URL has the following format:

https://<hostname>/#<project_room>/dataCollection/y.dataset.<key>

The <key> parameter is based on the selected display name of the data set and should be readable. Use this key in all your API requests to this specific data set.

Example

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

1.4.2 Path Section: Ingestion cycle

Several endpoints are used in the context of a specific ingestion cycle, such as commit data upload cycle (page 20), return ingestion cycle state (page 20), and cancel ingestion cycle (page 19). Because of this, they all include the following section:

/ingestionCycles/{ingestionCycle}

The **{ingestionCycle}** parameter refers to the technical key of the ingestion cycle and uses that value at runtime. Such a key is generated every time a new ingestion cycle is created. It is returned as value of the key property of the cycle after it is created or when all existing ingestion cycles are retrieved from the system using the API.

1.4.3 Retrieve API version

Retrieves the current API version of ARIS Process Mining.

GET /api/pub/dataIngestion/version

Output: Current API version

1.4.4 Retrieve source table definitions

Retrieves the column structure (name, data type, format) of the specified source tables.

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

Input: Query parameter **'fqns'** to filter by fully-qualified names

Output: List of **SourceTableDefinition** objects

1.4.5 Create or replace source tables

Creates or replaces a source table in ARIS Process Mining.

It depends on the parameter **forceReplace**. When parameter forceReplace = true, the tables with the same identifier will be replaced and all previously stored data deleted.

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

Input:

- List of **SourceTableDefinition** objects
  - When creating a source table, the name, namespace, and columns are required. The other properties are optional.

    Note that "_ARIS" is not allowed as a namespace.
  - When replacing a source table,

    - specify the identifier of the existing table in the form of a key, fully qualified name, or name and namespace. If you specify several identifiers, the priority order is key > fully qualified name > name and namespace. Identifiers with low priority are ignored if an identifier with high priority is present.

    - All other properties (excluding the identifiers) are optional. If a property is not set, it re-uses the values of the existing table. Note that the columns of a table are set in the same property. To remove a column, omit the column from the body. To add a column, repeat the information of the existing columns and include the new column.

    Note that "_ARIS" is not allowed as a namespace.

- The query parameter **forceReplace** indicates whether to replace the existing source tables with the same identifier. When the tables are replaced, all previously stored source data is deleted. If the parameter is not set to true, requests that would require replacement (for example, include the identifier of an existing table) are rejected.

Output:

List of **SourceTableDefinition** objects based on newly created or replaced source tables

1.4.6 Update a source table definition

Updates a source table definition, but leaves the existing data unchanged. It can be used to

- change the fully qualified name, namespace, and/or name.

  Note that "_ARIS" is not allowed as a namespace.
- change the table type from regular table to incremental.
- (re-)define a merge key for an incremental table.
- Add columns to an incremental table.
- Configure the schema of subsequent data deliveries to no longer contain particular columns. Existing data will still include the column.

  Columns of a regular table cannot be changed with this method. Use the Create or replace source tables (page 16) endpoint instead.

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

Input:

SourceTableDefinition object

- The source table must already exist.
  - Existing data is preserved.
  - Therefore, not all changes to the definition are permitted.
  - The source table in the URL can either be a key or a fully qualified name.
  - All properties of the source table definition are optional. If the properties are not set, the values of the existing table are reused.

    Note the following.

    The columns of a table will be set as a whole, that is, to remove a column just omit it from the body. To add a column, repeat the information of the existing columns and include the new column.

    If both fully qualified name and name and/or namespace are provided in the body, the fully qualified name takes precedence.

Output:

List of **SourceTableDefinition** objects based on newly created or replaced source tables

Note that updating the data of an incremental table can involve changes to the schema of the source table.

1.4.7 Check if data set is ready for ingestion

The check confirms that the uploading of the data or the data load can start.

Note that you can check the readiness state for EITHER starting a data load OR uploading the data, but not for both at the same time!

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

Input: DataIngestionCycle containing

- either SourceTableDefinitions based on the existing, fully configured data sources to update. You can specify any identifier. The other properties are optional and will be ignored.
- or a Boolean flag that indicates to start a data load. Specifying a list of source table definitions and setting the Boolean flag to true is not supported at this time. If you must run an upload after a data load, you must perform those separately as two data ingestion cycles.

Output: IngestionReadyState. Note that the readiness check for a data load does not necessarily consider all existing validation issues. Even when the user interface shows some validations issues, the readiness check might report a successful check.

1.4.8 Create a new ingestion cycle

Creates a new ingestion cycle.

Note that an ingestion cycle is created EITHER for starting a data load OR uploading data, but not for both at the same time.

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

Input: DataIngestionCycle containing

- either SourceTableDefinitions based on the existing, fully configured data sources to update. You can specify any identifier. The other properties are optional and will be ignored.

- or a Boolean flag that indicates to start a data load. Specifying a list of source table definitions and setting the Boolean flag to true is not supported at this time. If you must run an upload after a data load, you must perform those separately as two data ingestion cycles.

You must specify the input in the response body. In the case of data load, you must set "dataLoadTriggered" to true in the response body as follows:

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

Output: New DataIngestionCycle

1.4.9 Cancel ingestion cycle

Deletes an existing ingestion cycle (identified by ingestion cycle key).

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

Output: Canceled DataIngestionCycle

1.4.10 Upload data

Uploads data to ARIS Process Mining for the specified source table(s) and automatically sorts the columns based on the column order in ARIS Process Mining.

Data must have the correct structure (number and order of columns with correct data types and formats).

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

Input:

- Source table identifier as path parameter. The identifier can be either a key or a fully qualified name.
- List of objects as body, representing the new source data entries.
  - The order of the columns corresponds to the order specified when creating the source table and returned by the GET operation on the sourceTableDefinitions.
  - The timestamp data can only be passed as strings formatted in the date and time format of the corresponding source table column.
  - Large sets of data can be uploaded over multiple requests. The data in each request is stored in a temporary form on the server side.

Output: Success result if the data was received without error.

1.4.11 Drop source table

Drops a specified source table (definition and content).

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

Input: Source table identifier as path parameter. The identifier can either be a key or a fully qualified name.

Output: Success result if deletion was performed without error.

1.4.12 Commit data upload cycle

Notifies ARIS Process Mining that the data upload is complete and the start of the ingestion in ARIS Process Mining begins.

When the upload in ARIS Process Mining completes without an error, the status of the ingestion cycle is updated to "COMPLETED_SUCCESSFULLY". This is a precondition for starting a new ingestion cycle for loading the data.

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

Output: Running DataIngestionCycle

1.4.13 Retrieve ingestion cycles

Retrieves all existing ingestion cycles for a data set.

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

Output: List of DataIngestionCycle objects

The endpoint is available from ARIS Process Mining version 10.18.

1.4.14 Return ingestion cycle state

Retrieves the state of the specified ingestion cycle.

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

Output: The state value is based on the state of the corresponding run log entry.

1.5 Data transfer objects (DTOs)

You can use the following data transfer objects (DTOs) for the data ingestion API.

1.5.1 SourceTableDefinition

As input

Only in list form, either standalone as presented here or as part of a data ingestion cycle (DataIngestionCycle) (see below). Properties can be mandatory or optional, depending on if a table should be created or replaced.

```
[
	{
		"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"
			}
		]
	}
]
```

As output

Only in list form, either standalone as presented here or as part of a data ingestion cycle (DataIngestionCycle) (see below).

```
[
```

```
	{
		"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"
			}
		]
	}
]
```

- Keys will be generated on the server.
- The fully qualified name (fullyQualifiedName) consists of the name and namespace, separated by '.'.
- The persistence mode (persistenceMode) can either be OVERWRITE or APPEND. See chapter Persistence mode (page 29) for more details.
- Columns can be of type DOUBLE, LONG, STRING, and FORMATTED_TIMESTAMP.
- The merge key (mergeKey) is optional. The merge key is only required for merging new data with existing data.

NOTE

When a source table definition is created or updated, an **_ARIS_lastChanged** column of **timestamp** type is always automatically added to the uploaded table. The **_ARIS_** prefix is reserved for internal use. You must not specify columns whose names begin with **_ARIS_**, and requests to create tables containing such a column name will fail.

When a source table definition is retrieved from the API, columns whose names begin with **_ARIS_** are omitted. If you updated a source table definition using the API (as opposed to replacing it) a subsequent **GET** will only return the columns configured by that update and omit the "removed" columns. However, these columns and their existing data are of course still present and are still processed within ARIS Process Mining. This is to ensure that an API client always receives the table in its most recent form (using its most recent schema), and contains only columns that are known to it.

1.5.2 DataIngestionReadyState

Used only as output after a readiness check. In case of 'not ready', the ready property is set to 'false' and the object contains a cause with code and message.

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

Causes consist of a code and a message to indicate why exactly the data set is not ready. The code is four digits long and always prefixed with "INR" for "Ingestion - Not Ready". A list with the concrete codes and the semantics can be found in the table below.

Code	Semantic
INR1000	Undefined. Used for unexpected situations.
INR1001	Data set in process. The referenced data set is locked by another process (data ingestion cycle, manual data load, etc.). The user must wait until the locked is lifted and repeat the request.

Code	Semantic
INR1002	Living process quota exceeded. Usage of the data ingestion API is limited to data sets with a 'living process' license. The assigned 'living process' license defines an upper bound (process quota) for the number of process instances within the corresponding data set. Code INR1002 indicates that the process quota defined by the assigned 'living process' license is already exceeded and it is not allowed to ingest more data. The user can reduce the number of process instances within the dataset by deleting process instances or alternatively assign a 'living process' license with a higher process quota.
INR1003	Unexpected source table type. Indicates that at least one of the referenced source tables has an unexpected type, for example, was created via the CSV upload. To use a table in the context of the data ingestion API, it either needs to have been created by that API or updated by it via the replace source table functionality.
INR1004	No data to load. Indicates that there is no new pending data for the tables referenced in the data modeling section. Therefore a data load is not necessary.

1.5.3 DataIngestionCycle

As input

In case of data upload

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

In case of data load

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

As output

Either in list form


```
[
    {
        "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"
        }
    }
]
```


or standalone after creation, update, or cancelation, for example.

In case of data upload

```
{
    "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"
    }
}
```


In case of data load


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

1.5.4 DataIngestionCycleState

Used only as output, standalone or as part of a data ingestion cycle (DataIngestionCycle) (see above). Possible states are: ACCEPTING_DATA, INGESTING_DATA, COMPLETED_SUCCESSFULLY, CANCELED, and FAILED.

In case of 'FAILED', it will return a cause. Causes consist of a code and a message to indicate what happened exactly. The code is four digits long and always prefixed with "IER" for "Ingestion - Error".

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

TABLEDATA

Used only as input for data upload. Values must conform to the schema of the targeted source table. Null is a valid value.

```
[
    [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

Used only as part of a source table definition (SourceTableDefinition) (see above).

LONGCOLUMN

Used only as part of a source table definition (SourceTableDefinition) (see above).

DOUBLECOLUMN

Used only as part of a source table definition (SourceTableDefinition) (see above).

FORMATTEDTIMESTAMPCOLUMN

Used only as part of a source table definition (SourceTableDefinition) (see above).

DEFAULTRESULT

Used only as standalone output, either when the operation performed does not have a dedicated result object itself (deletion of a source table, upload of source data) or when an

error occurs on the server (any operation). The successful property of this object is either set to true or to false accordingly.

In case of false, the object will also contain a cause with a message.

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

APIVERSION

Used only as output after an API version check.

```
{
  "apiVersion": "3.2"
}
```

1.5.5 Authentication response

The response of your authentication request to the ARIS cloud is a JSON object that includes the tenant, a URL, and an access token.

{

"tenant": "<project_room>",

"token": "<access_token>,

"url": "<any_URL>"

}

Example

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

The response of your authentication request to the ARIS Enterprise cloud is a JSON object that includes an application token:

```
{
'applicationToken': '…'
}
```

1.6 Valuable information

1.6.1 Persistence mode

All source tables have a persistence mode that determines how the new data is processed on the server.

There are three different persistence modes:

OVERWRITE

This is the default setting to follow the standard behavior from previous versions. If this mode is set, the new uploaded data overwrites the existing data. The overwritten data is lost and cannot be recovered. If required, you must upload the overwritten data again.

APPEND (WITHOUT MERGE KEY)

If this mode is set, instead of overwriting the already persisted table data on the server, new uploaded data is appended to the existing data. The new data rows are added at the end in the order in which they are received. This causes the size of the existing source table to grow. Persisting old data a second time with this setting (as if they were new rows) will lead to duplicate entries. This can impact the accuracy of the analysis results.

Note that currently the only way to select this mode for a source table is to use the Data Ingestion API to either create a new table or replace an existing one.

APPEND WITH MERGE KEY

If this mode is set, the uploaded data is merged with already persisted table data on the server. In merge mode, new data is appended to the table in additional rows and existing rows of the table will be overwritten individually only if the corresponding row of the uploaded table is more recent. At the end of the data upload, the source table contains all the data.

The ingestion API uses a merge key to merge existing data and new data in the source table. You can use the merge mode by setting the merge key in the source table definition (page 21). You also must set the persistence mode (persistenceMode) to **APPEND** in the source table definition.

To configure a merge key for an existing table, you must send a source table definition with the new merge key to the server. To do this, you can use the Create or replace source tables (page 16) endpoint. The merge key is automatically added to a source table when the source table is created or replaced (page 16).

NOTE

A source table with the OVERWRITE persistence mode corresponds to a standard table in ARIS Process Mining and a source table with the APPEND consistence mode and with a specified merge key corresponds to an incremental table accordingly.

1.6.2 Limits

REQUEST SIZE

The maximum accepted size of a request to create or update data is limited to 100 MB. If this maximum is exceeded, the request is rejected. If you want to create or update more data, split the data into multiple requests.

SOURCE TABLES

TOTAL NUMBER OF SOURCE TABLES

The maximum number of source tables that can be created using the data ingestion API is 100. Each time new tables are created using the API, a check verifies if the maximum number is exceeded. If the number is exceeded, the corresponding request is rejected. All existing tables are counted towards the maximum allowed number, regardless of their origin (API, extraction, manual file upload). This limit does not affect the replacement of the source table.

NUMBER OF SOURCE TABLES PER REQUEST

The maximum number of source tables that can be created with one request is 50. If this number is exceeded, the corresponding request is rejected.

NUMBER OF COLUMNS

The maximum number of columns that can be created for a source table using the data ingestion API is 500. If this number is exceeded, the corresponding request is rejected. This limit affects both the creation and replacement of the source table.

TOTAL NUMBER OF TASKS

The maximum number of tasks (including ingestion cycles) that can be maintained simultaneously is 350. Each time a new cycle is created using the API, a check verifies if the maximum number is exceeded. If the number is exceeded, the corresponding request is rejected. All existing tasks that are still maintained count towards the accepted maximum, independent of their type (ingestion cycle, extraction, manual file upload, data load, recalculation, process data deletion) or origin (API, automation, manual execution). Maintained tasks are cleaned up automatically at regular intervals of 30 minutes. The cleanup routine deletes all completed tasks, except the 250 most recent entries.

UPLOADS

NUMBER OF DATA UPLOAD TARGETS

The maximum number of source tables (data upload targets) that can be referenced by one upload cycle of the data ingestion API is 100. If this maximum is exceeded, the corresponding

request is rejected. If there are more source tables in need of an upload, they must be split into multiple upload cycles.

NUMBER OF PENDING DATA PACKAGES

When uploading data using the data ingestion API, the allowed maximum number of pending upload data packages per table is 50. If this number is exceeded, the corresponding request is rejected. If more data should get uploaded to the target table, set the containing ingestion cycle to completed, which starts the server-side persistence. After the persistence (and the ingestion cycle) is completed, a new cycle can be created to upload the remaining data. Note that the second upload cycle should only be started immediately if the persistence mode of the target table is set to APPEND. If the mode is set to OVERWRITE, you must first load the data (load cycle). Only after loading the data, the remaining data can be uploaded safely.

1.7 webMethods.io connector for ARIS Process Mining

The webMethods.io connector for ARIS Process Mining uses the data ingestion API for transferring data from any data source to ARIS Process Mining. With the webMethods.io connector for ARIS Process Mining, you can, for example, create a table, upload data to the created table, and trigger a data load operation in ARIS Process Mining.

Predefined operations allow you to directly use the most common REST resources and operations or reduce the complexity of customizing REST operations.

For details on using the webMethods.io connector for ARIS Process Mining, please see the webMethods.io documentation.

The following list contains all predefined operations that are provided by the webMethods.io connector for ARIS Process Mining and shows which data ingestion endpoints (page 15) they refer to.

Operation	Reference to endpoints
Retrieve API Version	Retrieve API version (page 15)
Retrieve Source Table Definitions	Retrieve source table definitions (page 16)
Create Source Tables	Create or replace source tables (page 16)
Replace Source Tables	Create or replace source tables (page 16)
Update a source table definition	Update a source table definition (page 17)
Is Ready for Data Upload	Check if data set is ready for ingestion (page 18)
Is Ready for Data Load	Check if data set is ready for ingestion

Operation	Reference to endpoints
	(page 18)
Create Data Upload Cycle	Create a new ingestion cycle (page 18)
Start Data Load	Create a new ingestion cycle (page 18)
Retrieve Cycle State	Retrieve ingestion cycles (page 20)
Cancel Cycle	Cancel ingestion cycle (page 19)
Commit Data Upload Cycle	Commit data upload cycle (page 20)
Drop Source Table	Drop source table (page 20)
Retrieve Cycles	Retrieve ingestion cycles (page 20)
Upload Data	Retrieve source table definitions (page 16) Upload data (page 19)

2 Support and legal information

This section provides you with some general information regarding product support and legal aspects.

2.1 Documentation scope

The information provided describes the settings and features as they were at the time of publishing. Since documentation and software are subject to different production cycles, the description of settings and features may differ from actual settings and features. Information about discrepancies is provided in the Release Notes that accompany the product. Please read the Release Notes and take the information into account when installing, setting up, and using the product.

If you want to install technical and/or business system functions without using the consulting services provided by Software GmbH, you require extensive knowledge of the system to be installed, its intended purpose, the target systems, and their various dependencies. Due to the number of platforms and interdependent hardware and software configurations, we can describe only specific installations. It is not possible to document all settings and dependencies.

When you combine various technologies, please observe the manufacturers' instructions, particularly announcements concerning releases on their Internet pages. We cannot guarantee proper functioning and installation of approved third-party systems and do not support them. Always follow the instructions provided in the installation manuals of the relevant manufacturers. If you experience difficulties, please contact the relevant manufacturer.

If you need help installing third-party systems, contact your local Software GmbH sales organization. Please note that this type of manufacturer-specific or customer-specific customization is not covered by the standard Software GmbH software maintenance agreement and can be performed only on special request and agreement.

2.2 Data protection

Software GmbH products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR).

Where applicable, appropriate steps are documented in the respective administration documentation.

2.3 Support

If you have any questions on specific installations that you cannot perform yourself, contact your local Software GmbH sales organization (https://www.softwareag.com/corporate/company/global/offices/default.html). To get detailed information and support, use our Web sites.

If you have a valid support contract, you can contact **Global Support ARIS** at: **+800 ARISHELP**. If this number is not supported by your telephone provider, please refer to our Global Support Contact Directory.

For issues regarding the product documentation, you can also send an e-mail to documentation@softwareag.com (mailto:documentation@softwareag.com).

ARIS COMMUNITY

- Download products, updates and fixes
- Find information, expert articles, issue resolution, videos, and communication with other ARIS users

If you do not yet have an account, register at ARIS Community**.**

PRODUCT TRAINING

You can find helpful product training material on our Learning Portal.

TECH COMMUNITY

You can collaborate with Software GmbH experts on our Tech Community Web site. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software GmbH news and announcements.
- Explore our communities.

- Go to our public GitHub and Docker repositories and discover additional Software GmbH resources.

PRODUCT SUPPORT

Support for Software GmbH products is provided to licensed customers via our Empower Portal (https://empower.softwareag.com/). Many services on this portal require that you have an account. If you do not yet have one, you can request it. Once you have an account, you can, for example:

- Add product feature requests
- Search the Knowledge Center for technical information and tips
- Subscribe to early warnings and critical alerts
- Open and update support incidents.