syncsort

DMX

REST API Reference Guide

Lineage and management server web APIs

Version 9.7.3

DMX REST API Reference Guide

Version 1.0

Last Update: 01 March 2021

Contents

Introduction

The DMX REST API provides web-based request support for several DMX features. If you have a network connection to the DMX management service and the URL for the HTTP server, you can use DMX REST resources to access the following:

- Job Status
- Logging
- Lineage data

The responses are in JSON format, and the fields presented in this guide represent the minimum supported models. There may be additional properties included/generated in the responses but additional data should be ignored, as undocumented fields are not supported.

DMX REST requests

To make a request, use a REST-enabled markup or platform to form requests containing the following elements:

- An HTTP method: GET, POST, PUT, or DELETE.
- A resource identifier (URI)
- One or more parameters

Refer to the resource description and examples below.

REST API URI Syntax

The REST API URI prefix always contains the web server host name, port number, and web root of the REST server. The default web server is the host where DMX is installed. The default port for HTTP/HTTPS REST services is 8280.

The remaining resource identifiers for a DMX REST URI has the following syntax:

```
/api/v1/history/<resource>/[/{identifier}]/<subresource>/
    [?<parameter=value>{&<parameter=value>}]
```

- `/api/v1/history` is the default web root and directory for REST API resources
- `<resource>` is the name of the REST API resource.
- `{identifier}` is a parameter to a branch in the REST API resource.
- `<subresource>` is the name of a branch in the REST API resource.
- `<parameter=value>` specifies the parameters and values that a GET request requires.

Authentication

To successfully submit a REST API request, a requesting client must be authenticated by the DMX management service, dmxmgr. The authentication method depends on how the management service is setup for User Authentication, either simple or LDAP. Please refer to the DMX Help for instructions on DMX management server installation and configuration.

REST API Resources

DMX REST resources are abstractions of DMX functionality, such as a query or a job status request. Table 1-1 lists the REST API resources and the operations associated with the HTTP methods used by each resource.

Table 1-1: DMX REST API Resources

Resource	Description	HTTP methods
projects/dmxjob/	Returns the unique ID for a job	GET
projects/<jobRunId>/ lineage/	Accesses lineage data (entire report)	GET
projects/<jobRunId>/ lineage/entities/	Accesses lineage data entities	GET
projects/<jobRunId>/status	Returns the status for a job or schedule	GET
projects/<jobRunId>/log	Accesses job run logs	GET, DELETE

Job ID

Every DMX job is tracked by a unique identifier, whether scheduled or run. To access metadata about a job, you require its unique identifier, and the projects search API retrieves job run IDs that match various parameters.

URI

```
projects/dmxjob/
```

HTTP Methods

GET

Parameters

All parameters, shown in Table 1-2, are optional. If no parameters are specified, every job ID in the system's history is retrieved.

Table 1-2: Job ID search parameters

Parameter	Optional?	Description
name=<jobName>	Optional	Short job file name.
jobDirectory=<dir>	Optional	The job path on the node from which the job was submitted.

Parameter	Optional?	Description
user=<user>	Optional	The authorizing user for a job.
startTime=<time>& endTime=<Time>:	Optional	A range of job start times. Times must be in "`yyyy-MM-ddT hh:mm:ss`" (ISO 8601) format or an error is reported.

Response

Text containing the unique job identifier. If there are no parameters, a list of all jobs is retrieved.

Lineage Reports

A complete list of the lineage data is available from the lineage API resource.

URI

```
projects/<jobRunId>/lineage/
```

HTTP Methods

GET

Parameters

The parameters for lineage full reports are shown in table 1-3.

Table 1-3: Lineage query parameters

Parameter	Optional?	Description
<jobRunId>	Required	Unique identifier for job run.
ids=<entityId1, entityId2, ...>	Optional	Filter for a specific entityIds to retrieve. Entities in the lineage that do not match a value listed in entityIds won't appear in the response.
direction=<int>	Optional	Lineage directions, which apply for each entity retrieved. Values must be: • 1: downstream lineage entities • -1: upstream lineage entities • 0: both upstream and downstream lineages

Response

Entity data is delivered in lineage JSON format, which has the following structure:

```
{
      "entities": {
            "entityid": {
                  "field":"value",
```

```
                        ...
                }
        }
        "relations": [{
                        "from":"value",
                        "to":"value",
                },
                ...
        ]
}
```

Field attributes for the response are shown in table 1-4.

Table 1-4: Lineage JSON response fields

Field	Description
entities	Mandatory structure containing a list of all the lineage entities
entityId	Mandatory string that is a unique identifier for each entity. The "entityId" value is dependent on the entity category and contents. See Lineage Entities below for entity structures.
relations	Optional array of directed symbolic links in the entity graph, which indicate lineage order. • "from" value is the entityId for the origin of a link • "to" value is the entityId for the target of a link

Example

The following lineage request retrieves the downstream lineage for entity 4AD32679DDDE3B77EB0C33F9A5910837 from job c1ea4f4a-8e60-468c-b517-cbe7a4fe0d84:

```
https://localhost:8280//api/v1/history/projects/c1ea4f4a-8e60-468c-
b517-cbe7a4fe0d84/lineage/
?ids=[4AD32679DDDE3B77EB0C33F9A5910837]&direction=1
```

The response is:

```
{
    "entities": {
        "BEAB9CBDA58B690B5FA0C53EB5235657": {
            "category": "PROCESS",
            "name": "convertNames",
            "properties": {
                "type": "TASK",
                "path": "C:\\tmp\\lineage\\api_demo\\convertNames.dxt"
            }
        },
        "ECB5938AFA15AE877E20BBE76087108F": {
            "category": "DATASET",
            "name": "sourcePeople.txt",
            "properties": {
                "type": "FILE",
```

```
                "scheme": "FILE",
                "host": "FARAG-N-T3600",
                "path": "C:\\tmp\\lineage\\api_demo\\sourcePeople.txt"
            }
        },
        "94ACAC772C09A81231734E7F09011EA9": {
            "category": "DATASET",
            "name": "targetPeople.txt",
            "properties": {
                "type": "FILE",
                "scheme": "FILE",
                "host": "FARAG-N-T3600",
                "path": "C:\\tmp\\lineage\\api_demo\\targetPeople.txt"
            }
        },
        "556b7e758f320d611a0d533a22f6583c": {
            "category": "TRANSFORMATIONGROUP",
            "name": "trimLastName = Trim( source.lastName ) -->
                cleanName = trimFirstName + ' ' + trimLastName -->
                cleanName",
            "parent": "BEAB9CBDA58B690B5FA0C53EB5235657",
            "properties": {
                "transformations": [
                    {
                        "name": "trimLastName =
                            Trim( source.lastName )",
                        "type": "VALUE"
                    },
                    {
                        "name": "cleanName =
                            trimFirstName + ' ' + trimLastName",
                        "type": "VALUE"
                    },
                    {
                        "name": "cleanName",
                        "type": "REFORMAT ITEM"
                    }
                ]
            }
        },
        "b8a4d420c1219c1d77f41a7b7cad287e": {
            "category": "TRANSFORMATIONGROUP",
            "name": "trimLastName = Trim( source.lastName ) -->
                trimLastName",
            "parent": "BEAB9CBDA58B690B5FA0C53EB5235657",
            "properties": {
                "transformations": [
                    {
                        "name": "trimLastName =
                            Trim(source.lastName )",
                        "type": "VALUE"
                    },
                    {
                        "name": "trimLastName",
                        "type": "REFORMAT ITEM"
                    }
                ]
```

```
            }
        },
        "C90DE343F121E43F32FFC48919DB2D5E": {
            "category": "DATAFIELD",
            "name": "source.lastName",
            "parent": "ECB5938AFA15AE877E20BBE76087108F",
            "properties": {}
        },
        "D98964D6EAB651B57C423B3146409BA9": {
            "category": "DATAFIELD",
            "name": "target.fullName",
            "parent": "94ACAC772C09A81231734E7F09011EA9",
            "properties": {}
        }
    },
    "relations": [
        {
            "from": "C90DE343F121E43F32FFC48919DB2D5E",
            "to": "556b7e758f320d611a0d533a22f6583c"
        },
        {
            "from": "556b7e758f320d611a0d533a22f6583c",
            "to": "D98964D6EAB651B57C423B3146409BA9"
        },
        {
            "from": "C90DE343F121E43F32FFC48919DB2D5E",
            "to": "b8a4d420c1219c1d77f41a7b7cad287e"
        },
        {
            "from": "b8a4d420c1219c1d77f41a7b7cad287e",
            "to": "D158BDE35681F5BBAECA86A460D9D0A1"
        }
    ]
}
```


Lineage Entities

Lineage data is logically segmented into the different entities that stored, moved or transformed the data while a job runs. The lineage entities resource accesses information identified by entity.

URI

```
projects/<jobRunId>/lineage/entities/
```

HTTP Methods

GET

Parameters

The parameters for entities support most properties as a query, as shown in table 1-5.

Table 1-5: Lineage entity query parameters

Parameter	Optional?	Description
<jobRunId>	Required	unique identifier for job run.

Parameter	Optional?	Description
query=<property:value>	Optional	filter for a specific or wildcard property value.
query=<id:entityId>	Optional	filter for a unique entity id within the job lineage

NOTE: `query="*"` returns all entities.

Response

Entity data is delivered in lineage JSON format, which has the following structure:

```
"entityId": {
"category": "value",
      "name": "value",
      "parent": "value"
      "properties": {
            "field":"value",
            ...
      }
}
```

Field attributes for the response are shown in table 1-6.

Table 1-6: Lineage entity JSON response fields

Field	Description
entityId	mandatory string that is the unique identifier for this entity.
category	mandatory string, which is one of the supported category enumerated values: • DATASET • DATEFIELD • PROCESS • TRANSFORMATIONGROUP
name	mandatory string that is the displayed name of the entity.
parent	optional string for the parent entity's unique identifier. Datasets and top-level jobs do not have parents.
properties	mandatory structure, which is formatted based on the "category" value. See the following sections for the full property structures. • DATASET "type" is a mandatory string, which is one of the supported category enumerated values: o FILE o TABLE

Field	Description
	○ SALESFORCE ○ SAP ○ MQ ○ FILE-BASED ○ STDIN/STDOUT ○ PIPE • DATEFIELD • PROCESS • TRANSFORMATIONGROUP

DATASET type:FILE Properties

The response JSON format for Properties when the category is DATASET and the type is FILE is:

```
"properties":{
      "scheme": "value",
      "hadoopFilesystemScheme": "value",
      "host": "value",
      "port": "value",
      "path": "value",
      "fileType": "value",
      "recordLength": "value",
      "recordAlignment": "value",
      "encoding": "value",
      "isCompressed": "true|false",
      "isLanded": "false",
      "isDirectory": "true|false",
      "matchedFiles": [
            "value",
            ...
      ]
}
```

Field attributes for the properties block are:

Table 1-7 JSON fields for DATASET type:FILE Properties

Field	Description
scheme	Mandatory string, which is one of the supported scheme enumerated values: • FILE • HADOOP • S3 • GS • FTP • FTPS • SFTP • CONNECTIDIRECT

Field	Description
hadoopFilesystemScheme	Optional string, only available if the scheme is HADOOP with one of the supported hadoopFileSystemScheme enumerated values: • HDFS (default) • S3 • S3N • S3A • GS
host	Optional string for the host/server/bucket where the file resides or is accessed
port	Optional string for the port assigned to the data service that provides access to the file
path	Optional string for the absolute path to the file
fileType	Optional string, only available for stream DATASETS. Possible values are the same as displayed in GUI.
recordLength	Optional string, only available for fileTypes whose format is fixed length, for the byte length of the record.
recordAlignment	Optional string, only available for stream DATASETS. Possible values are the same as displayed in GUI"
encoding	Optional string; valid values are the same as displayed in GUI
isCompressed	Optional string true/false
isLanded	Optional string Boolean "false"
isDirectory	Optional string Boolean true/false: true if the file is a directory without wildcard, false is the default value
matchedFiles	Optional array of strings: a list of file names expanded from the dataset file name, which has a wildcard or directory name

DATASET type:TABLE Properties

The response JSON format for Properties when the category is DATASET and the type is TABLE is:

```
"properties": {
      "origin": "value",
      "scheme": "value",
      "host": "value",
      "port": "number",
```

```
            "database": "value",
            "schema": "value",
            "table": "value",
            "sqltext": "value",
            "sqltextTables [{
                        "schema": "value",
                        "table": "value",
                  },
                  ...
            ]
      }
```

Field attributes for the properties block are:

Table 1-8 JSON fields for DATASET type:TABLE Properties

Field	Description
origin	Mandatory string, which is one of the supported origin enumerated values: • DB2 • ORACLE • SYBASE • TERADATA • ODBC • JDBC • CHANGEDATACAPTURE If JDBC, the scheme, host, port, and database is derived from the JDBC URL string. If origin is not JDBC, the database is set, but scheme, host, and port are not set.
scheme	String, mandatory if origin is JDBC. Omitted otherwise.
host	String, mandatory if origin is JDBC, for the host where the DSN is defined for the JDBC connection. Omitted otherwise.
port	Optional string if origin is JDBC. Omitted otherwise.
database	Mandatory string for the database containing the table. If origin is ODBC, this is the DSN.
schema	Optional string for the table schema, if available.
table	String, mandatory if sqltext is empty string, for the unqualified table name, if available.
sqltext	String, mandatory if table is empty string -- the sqltext if available

Field	Description

sqltextTables	Optional array of structures, defined as:
	`{`
	`"schema": "value",`
	`"table": "value"`
	`}`
	Where
	• **Schema**: mandatory string for the table schema. Can be empty.
	• **Table**: mandatory string for the table name.

DATASET type:SALESFORCE Properties

The response JSON format for Properties when the category is DATASET and the type is SALESFORCE is:

```
"properties": {
      "database": "value",
      "table": "value",
      "sqltext": "value"
}
```

Field attributes for the properties block are:

Table 1-9 JSON fields for DATASET type:SALESFORCE Properties

Field	Description
database	Mandatory string for the Salesforce connection URL.
table	String, mandatory when soql text is absent, for the name of the Salesforce object when the source/target is a Salesforce object.
sqltext	String, mandatory when object is absent, for the name of the Salesforce object when the source/target is SOQL text

DATASET type:SAP Properties

The response JSON format for Properties when the category is DATASET and the type is SAP is:

```
"properties": {
      "system": "value",
      "client": "value",
      "module": "value"
},
```

Field attributes for the properties block are:

Table 1-10 JSON fields for DATASET type:SAP Properties

Field	Description
system	Mandatory string for the SAP system name
client	Mandatory string for the SAP client name
module	Mandatory string for the SAP module

DATASET type:MQ Properties

The response JSON format for Properties when the category is DATASET and the type is MQ (message queue) is:

```
"properties": {
      "origin": "value",
      "stream": "value",
      "table": "value"
},
```

Field attributes for the properties block are:

Table 1-11 JSON fields for DATASET type:MQ Properties

Field	Description
origin	Mandatory string, which is one of the supported origin enumerated values: • APACHEKAFKA • MAPR • WEBSPHERE
database	Mandatory string for the kafka server, mapr url, or websphere queue manager
stream	String, mandatory if origin is MAPR, for the mapr stream
table	Mandatory string for the kafka or mapr topic, or the websphere queue

DATASET type:FILE_BASED, STDIN/STDOUT, or PIPE Properties

The response JSON format for Properties when the category is DATASET and the type is FILE_BASED, STDIN/STDOUT, or PIPE is:

```
"properties": {
      "isLanded": "false",
      "fileType": "value"
      "encoding": "value",
      "isCompressed": "true|false"
```

```
}
```

Field attributes for the properties block are:

Table 1-12 JSON fields for DATASET type:FILE_BASED, STDIN/STDOUT, or PIPE Properties

Field	Description
isLanded	Mandatory string Boolean that is always false
fileType	Optional string, only available for stream DATASETS, with the same possible values as those displayed in GUI
encoding	Optional string, with the same possible values as those displayed in GUI
isCompressed	Optional string Boolean, either true or false

DATAFIELD Properties

The response JSON format for Properties when the category is DATAFIELD is:

```
"properties": {
      "originalColumnName": "value",
      "dataType": "value",
      "originalDataType": "value",
      "length": "valule",
      "originalLength": "value",
      "format": "value",
      "encoding": "value",
      "nullability": "value",
      "originalIsNullable": "true|false",
      "originalScale": "value"
},
```

Field attributes for the properties block are:

Table 1-13 JSON fields for DATAFIELD Properties

Field	Description
originalColumnName	Optional string for the name of the original table column when parent.properties.type is TABLE.
dataType	Optional string, with the same possible values as those displayed in GUI.
originalDataType	Optional string for the data type as defined by the 3rd party (e.g databases, xml, avro etc.), with the same possible values as those displayed in GUI.
length	Optional string for the field length in fixed length fields, and absent for variable length fields.

Field	Description
originalLength	Optional string for the original field length for fixed length fields as defined by the 3rd party.
format	Optional string for Number and Date/time dataTypes, with the same possible values as those displayed in GUI.
encoding	Optional string, with the same possible values as those displayed in GUI. If the parent DATASET has encoding specified, then the value is 'Inherited from source'.
nullability	Optional string, with the same possible values as those displayed in GUI. If 'Default task setting' is set, then the value is 'Inherited from Task settings'
originalIsNullable	Optional string Boolean “true” or “false”, available for applicable 3rd party fields, like database columns.
originalScale	Optional string available for applicable 3rd party fields, like database columns.

PROCESS Properties

The response JSON format for Properties when the category is PROCESS is:

```
"properties": {
      "type": "value",
      "path": "value",
      "occurrence": "value"
},
```

Field attributes for the properties block are:

Table 1-14 JSON fields for PROCESS Properties

Field	Description
type	Mandatory string, which is one of the supported type enumerated values: • JOB • TASK • CUSTOM TASK • EXTENDED TASK • READER
path	Mandatory string, with a format specific to the type field: • For jobs/tasks, the full path of the job/task • For custom task, the full command line with the exe and arguments

Field	Description
	• For extended task, 'Extended Task <xml name>, Arguments: <argument keyword> = <user specified value> [, ...]' • For reader, '<reader format, currently this is JSON|csv, only JSON is exposed in the UI>Reader(<stdin|source file>,<stdout|target file>)'
occurrence	Optional string when there are multiple occurrences of the PROCESS to make it unique. The value is a one-based number, with the default value of 1.

TRANSFORMATIONGROUP Properties

The response JSON format for Properties when the category is TRANSFORMATIONGROUP is:

```
"properties": {
      "transformations": [{
                  "name": "value",
                  "type": "enum",
                  "result": "value",
                  "arguments": [
                        "value",
                        ...
                  ]
            },
            ...
      ]
},
```

Field attributes for the properties block are:

Table 1-15 JSON fields for TRANSFORMATIONGROUP Properties

Field	Description

transformations	Mandatory array of structures, defined as:
	`{`
	`"name": "value",`
	`"type": "enum",`
	`"result": "value",`
	`"arguments": [`
	`"value",`
	`...`
	`]`
	`}`
	Where
	• **name**: Mandatory string for displayable text, typically "<type> <result> = <argument>".
	• **type**: Mandatory string, which is one of the supported transformation type enumerated values:
	o SORT KEY

Field	Description
	○ MERGE KEY ○ GROUPBY KEY ○ JOIN KEY ○ TARGET DATABASE MAPPING ○ REFORMAT CONDITION ○ REFORMAT MAPPING ○ REFORMAT ITEM ○ FILTER ○ VALUE ○ etc. • **result**: Optional string with the result of the transformation, if any. For example, the SORT KEY type transformation produces no result. • **arguments**: Optional array of string arguments. Valid arguments depend on the transformation type. For example, a REFORMAT CONDITION requires the source expression/field as an argument.

Example

The following URIs access single entity from job c1ea4f4a-8e60-468c-b517-cbe7a4fe0d84:

```
https://localhost:8280//api/v1/history/projects/c1ea4f4a-8e60-468c-
b517-cbe7a4fe0d84/lineage/entities/?query="name:.*field2"
```

Or

```
https://localhost:8280//api/v1/history/projects/c1ea4f4a-8e60-468c-
b517-cbe7a4fe0d84/lineage/entities/?query="id:
D07C395755DAB597413B170DB8C9F2C6"
```

The response is:

```
"D07C395755DAB597413B170DB8C9F2C6": {
      "category": "DATAFIELD",
      "name": "source.firstName",
      "parent": "ECB5938AFA15AE877E20BBE76087108F"
}
```

Job Status

For each job run, DMX assigns status messages. The status of a job indicates whether it completed, if there were error or exceptions, or when a job has paused, aborted, or been scheduled.

URI

```
projects/<jobRunId>/status
```

HTTP Methods

GET

Parameters

The `<jobRunId>` identifies which job's status to query

Response

Text, which is one of the strings shown in Table 1-16.

Table 1-16: Job status strings

Status String	Description
RUNNING	Job is running
SUCCEEDED	Job completed successfully without exceptions
EXCEPTIONS	Job completed successfully with exceptions
FAILED	Job failed after successful start
ABORTED	Job aborted after starting
NOT_PROCESSED	Job aborted before starting
FAILED_START	Job failed on startup
INTERRUPTED	Job interrupted due an unexpected or internal server error

Job Logs

As a job runs, DMX generates a log file. The job log API provides access to these job logs.

URI

```
projects/<jobRunId>/logs
```

HTTP Methods

GET, DELETE

Parameters

The `<jobRunId>` identifies which job's log to access

Response

text

Example

The following output is for a job that runs successfully, but one of the tasks produced an exception. When an exception occurs in a nested task or job, the exception is noted in the job and run status messages:

```
************************ BEGIN RUN ************************

Running job bce4adb0-a8d6-419c-867f-ee2207fdf3b4 ...
************************ BEGIN JOB perf20 ************************
```

```
************************ BEGIN TASK perf1 ************************
[DMX 18.5.2 Windows x86 32-bit Copyright (c) 2018 Syncsort Inc.]
[For license use by Syncsort Internal]
05/08/2018 13:08:24 - Processing
C:\tmp\lineage\api_demo\perf\perf1.dxt, last modified on 02/23/2018
13:28:47
05/08/2018 13:08:24 - DMX options validated. Processing continues.
DMX : (TERMMISSING) record terminator is missing for the last record of
source 1

                                DMX statistics

Source: C:\tmp\lineage\api_demo\perf\in1.txt
    last modified on 08/18/2017 12:56:41
Records read:                           1  Data read (bytes):
17,738
Records copied:                         1  Data copied (bytes):
17,738
Target: C:\tmp\lineage\api_demo\perf\out2.txt
    last modified on 05/08/2018 13:08:25
Records output:                         1  Data output (bytes):
7,779
Input record length:               17,738  Output record length:
7,779
Memory guideline from job (MB):       100
Virtual memory allocated (MB):         56  Physical memory used (MB):
81
Work space used (bytes):                0

Elapsed time:                  0:00:00.80  CPU time:
0:00:00.76

05/08/2018 13:08:25 - DMX has completed
************************ END TASK perf1   ************************

************************ BEGIN TASK perf2 ************************
[DMX 18.5.2 Windows x86 32-bit Copyright (c) 2018 Syncsort Inc.]
[For license use by Syncsort Internal]
05/08/2018 13:08:25 - Processing
C:\tmp\lineage\api_demo\perf\perf2.dxt, last modified on 02/23/2018
13:29:42
05/08/2018 13:08:25 - DMX options validated. Processing continues.

                                DMX statistics

Source: C:\tmp\lineage\api_demo\perf\out2.txt
    last modified on 05/08/2018 13:08:25
Records read:                           1  Data read (bytes):
7,779
Records copied:                         1  Data copied (bytes):
7,779
Target: C:\tmp\lineage\api_demo\perf\out3.txt
    last modified on 05/08/2018 13:08:26
Records output:                         1  Data output (bytes):
7,779
Input record length:                7,779  Output record length:
7,779
```

```
Memory guideline from job (MB):      100
Virtual memory allocated (MB):        56  Physical memory used (MB):
81
Work space used (bytes):               0

Elapsed time:                 0:00:00.77  CPU time:
0:00:00.74

05/08/2018 13:08:26 - DMX has completed
************************ END TASK perf2   ************************
Job has completed with exceptions.
Total elapsed time: 0:00:01.15
************************ END JOB perf20   ************************
Run has completed with exceptions.
************************ END RUN ************************
```