Metadata Extraction Guidance Document

# Introduction

This document contains step-by-step guidance in the process of developing a reliable solution to extract metadata from the IICS platform.  
Here you might be able to see the python code that calls two Rest-API’s providing one source data information side in the structure. Which stands for both Activity Log and IICS Objects Inventory. This way composing another table that makes visualizable any job ran related info. And the Snowflake data warehouse table provides the second source side of reliable data information in terms of load control specifications.

Index

[Introduction 1](#_Toc112866996)

[The Solution Pipeline/Process 1](#_Toc112866997)

[Python Code / Rest APIs 2](#_Toc112866998)

[Metadata Extraction Task-flow  2](#_Toc112866999)

[Data Model 6](#_Toc112867000)

[Snowflake Used Queries 7](#_Toc112867001)

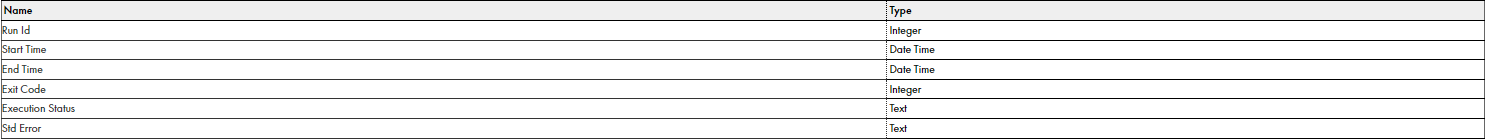
# The Solution Pipeline/Process

This solution was built and designed to automate the process of extracting metadata from IICS without the need for human hands in the process.

The Pipeline consists of IICS Task flow, CDI Mapping and Python Code.  
The Python Code is responsible for extracting the metadata from the IICS Repository for both Activity Log (for Mapping Task details) and Object Inventory (for Task flow).  
The python code uses Rest-API invocation to accomplish that. Once The python code extracts the metadata, it will be loaded into two tables in Snowflake (IICS\_TEMP tables) using the Truncate/Load approach to rewrite anything in the TEMPORARY tables.   
Once those IICS\_TEMP tables are loaded the IICS CDI Mapping will read those tables and the Load Control Setup/Task flow Tables. Then the CDI Mapping will relate that information and validate if the Operational Metadata already exists in the Target, before loading it in the Target table. The CDI Mapping uses only the Insert operation, and it does not truncate the target.  
  
**The object is to deliver Statistics aiming into each Mapping-task from each Execution-Plan. Only Operational Metadata for Completed Mapping Tasks is inserted in the Target table.  
The Snowflake table, IICS\_EDW\_LOAD\_DETAILS\_CTL will receive the metadata information.**

# Python Code / Rest APIs

As mentioned before…  
The python code is designed to call authorization for the IICS repository and then pull access to both Rest-APIs that give access in two log tables in the IICS.  
The Activities Log and the Object Inventory log.  
  
The code joins those tables to build information of each ran job breaking by two levels.  
Task-flow Level – Mapping-Task the working with Mapping-Task level.  
Providing output fields for use.  
Command source data paths:  
A picture containing application

Description automatically generated  
Command output fields:  
  
IICS repository limits the API to pull only 200 objects in the Repository Table. The first process used to timeout those registers.  
  
The solution was:

The last Python development change was done to pull only the last runtimes, this way filtering what we need by tagging task flows (information can be found by right-clicking each task flow).

Within all Main Execution Plan Task flows, Secondary Task flows, and LC Task fowls Tagged an endpoint filter was placed to only call Tagged ones in the Repository now pulling only the forty desired containing all their MTTs inside as stored information.

Metadata Extraction Task-flow  


The Task-flow is basically composed by:  
The python code command and the Metadata Extraction mapping task.  
Diagram

Description automatically generated

Once it starts.  
The command provides the output fields mentioned before and truncates the information.  
then the mapping task has the purpose of inserting metadata registers in the Snowflake Table.  
Mapping-Task / Mapping  




These items can be found by typing “metadata” in the IICS explorer tool.  
   
Graphical user interface, text

Description automatically generated

Diagram

Description automatically generated  
The mapping is composed by:  
2 sources,  
2 expressions,  
1 join,  
1 filter,  
1 lookup,  
1 sorter,  
1 sequencer and 1 target.

1. The **Sources**:

* Graphical user interface, application

  Description automatically generated   
  The first source is the LOAD\_CTL Snowflake Table:  
  SELECTED by a query a query sending 15 fields and converting ‘START\_TS’,’END\_TS’ and ‘LAST\_UPDATE\_DATE’ columns to date/time(TIMESTAMP) datatype:   
  SELECT IICS\_EDW\_LOAD\_TASKFLOW\_CTL.\*, IICS\_EDW\_LOAD\_SETUP\_CTL.EXECUTION\_PLAN\_NAME FROM

IICS\_EDW\_LOAD\_TASKFLOW\_CTL,

IICS\_EDW\_LOAD\_SETUP\_CTL

WHERE

IICS\_EDW\_LOAD\_TASKFLOW\_CTL.LOAD\_STATUS = 'C' AND IICS\_EDW\_LOAD\_TASKFLOW\_CTL.START\_TS >= DATEADD( day,-7,CURRENT\_TIMESTAMP) AND

IICS\_EDW\_LOAD\_SETUP\_CTL.LOAD\_STATUS = 'C' AND IICS\_EDW\_LOAD\_SETUP\_CTL.START\_TS >= DATEADD( day,-7,CURRENT\_TIMESTAMP) AND

IICS\_EDW\_LOAD\_TASKFLOW\_CTL.PLAN\_RUN\_ID = IICS\_EDW\_LOAD\_SETUP\_CTL.PLAN\_RUN\_ID AND

IICS\_EDW\_LOAD\_TASKFLOW\_CTL.CTL\_LOAD\_ID = IICS\_EDW\_LOAD\_SETUP\_CTL.CTL\_LOAD\_ID

ORDER BY

IICS\_EDW\_LOAD\_TASKFLOW\_CTL.TASKFLOW\_NAME

* Text

  Description automatically generated  
   The second source is a SELECTION of the two TEMP tables that are automatically truncated every ran of the python process. Called by the command that feds the TEMP tables every time the “TSKF\_LOAD\_METADATA\_EXTRACT\_CTL” runs in IICS. Feeding the Snowflake TEMP tables fields.  
  The Query:  
  SELECT \* FROM "EDW\_DEV"."EDW\_LOAD\_CT"."IICS\_TMP\_MTT\_METADATA" A, "EDW\_DEV"."EDW\_LOAD\_CT"."IICS\_TMP\_TSKF\_METADATA" B WHERE

A.MTT\_PARENT\_TASK\_FEDERATED\_ID = B.TSKF\_FEDERATED\_ID AND

A.MTT\_STATUS IN ('1','2') ORDER BY B.TSKF\_NAME

1. The first **Expression**:  
    Text

   Description automatically generated

* This expression was placed to replace the ‘TSFK\_NAME’ string column. Converted into a parsed-string by the next code:  
  ReplaceStr(0,

ReplaceStr(0, TSKF\_NAME, 'Enterprise Data Warehouse/EDW Taskflows/', ''),'Enterprise Data Warehouse/EDW Secondary Taskflows/','')

1. The **Join**:  
   Text

   Description automatically generated

* This joiner has the purpose of connecting all the Execution Plan information from LOAD CONTROL to the IICS Operational Metadata Information. Generating the ‘EXP\_SET\_EXCTRACT\_TASKFLOW\_NAME’ field.

1. The **Filter** and the **Lookup** work both together:  
   Graphical user interface, text, application, chat or text message

   Description automatically generated

* This filter will not allow already existing rows in the target and any cartesian information that generated false positives.
* Filter Query:  
  (MTT\_START\_TIMESTAMP > START\_TS AND MTT\_END\_TIMESTAMP < END\_TS)

AND

(ISNULL(:LKP.LKP\_CHECK\_METADATA\_EXISTS(TASKFLOW\_LOAD\_ID, CTL\_LOAD\_ID, TASKFLOW\_NAME, MTT\_NAME, PLAN\_RUN\_ID, TSK\_RUN\_ID, EXECUTION\_PLAN\_NAME) ))

* This lookup uses the LOAD\_CT table as the source and was made to verify if the Metadata record has been already inserted previously.
* Validating:  
  If the Metadata record exists in the target, it will not go downstream to the target. If the Metadata record does not exist, the record will be inserted into the target.

1. The second **Expression**:  
   Graphical user interface, text, application, Word

   Description automatically generated

* This expression generates a new field named ‘LAST\_UPDATE\_DATE\_exec’ as date/time.
* Using only IICS command SESSSTARTTIME code value.  
  

1. The next ETL is a **Sorter:**Text

   Description automatically generated

* This sorter sorts information ascending to lineup records by start date based in the fields “START\_TS” and “MTT\_START\_TIMESTAMP”. Table

  Description automatically generated

2. The **Target**: DETAILS\_CTL table.

* The target is basically the Load-Control Snowflake Table("EDW\_DEV"."EDW\_LOAD\_CT"."IICS\_EDW\_LOAD\_DETAILS\_CTL”in DEV and "EDW\_QA"."EDW\_LOAD\_CT"."IICS\_EDW\_LOAD\_DETAILS\_CTL” in Q.A.
* For this table to be completed all Main and Secondary Task flows must be run previously then hit RUN in the.
* The information in the table must be finished at Taskflow 25 according to all Task Flows and start at the Secondary ordered by Task flow Name.  
  First Secondary Task flow bellow:  
    
  

Last Main Execution Plan Task flow (Task flow 25) bellow:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

# Data Model

The Snowflake tables will be working in this solution feeding the IICS\_EDW\_LOAD\_DETAILS\_CTL table. This table is a foreign key (FK) at the IICS\_EDW\_LOAD\_\_TASKFLOW\_CTL table, an object that in turn is an FK in IICS\_EDW\_LOAD\_SETUP\_CTL. Text

Description automatically generated with low confidence

This way. “IICS\_EDW\_LOAD\_DETAILS\_CTL” will receive the “MTT\_NAME”,”MTT\_RUN\_START\_TIME” and “MTT\_RUN\_END\_TIME”.

# Snowflake Used Queries

**QUERY VALIDATE**  
  
SELECT \* FROM EDW\_DEV.EDW\_LOAD\_CT.IICS\_EDW\_LOAD\_TASKFLOW\_CTL  
WHERE IICS\_EDW\_LOAD\_TASKFLOW\_CTL.LOAD\_STATUS = 'C' AND IICS\_EDW\_LOAD\_TASKFLOW\_CTL.START\_TS >= DATEADD( day,-7,CURRENT\_TIMESTAMP)

ORDER BY  
TASKFLOW\_NAME

SELECT (CURRENT\_TIMESTAMP)

FROM DUAL TABLE

**COLUMNS ADDED IN IICS\_EDW\_LOAD\_DETAILS\_CTL**

ALTER TABLE "EDW\_DEV"."EDW\_LOAD\_CT"."IICS\_EDW\_LOAD\_DETAILS\_CTL"  
ADD COLUMN PARENT\_TSK\_RUN\_ID VARCHAR(1024),  
ADD COLUMN TSK\_RUN\_ID VARCHAR(1024),  
ADD COLUMN EXECUTION\_PLAN\_NAME VARCHAR(1024);

**COLUMNS ADDED IN IICS\_EDW\_LOAD\_DETAILS\_CTL**

ALTER TABLE "EDW\_DEV"."EDW\_LOAD\_CT"."IICS\_EDW\_LOAD\_DETAILS\_CTL"  
ADD COLUMN MTT\_RUN\_START\_TIME TIMESTAMP  
ADD COLUMN MTT\_RUN\_END\_TIME TIMESTAMP

**COLUMNS DROPED IN IICS\_EDW\_LOAD\_DETAILS\_CTL**

ALTER TABLE "EDW\_DEV"."EDW\_LOAD\_CT"."IICS\_EDW\_LOAD\_DETAILS\_CTL" DROP COLUMN  
PROJECT\_NAME,  
INTEGRATION\_ID,  
ETL\_PROC\_WID;

**QUERY TO IICS MAPPING TASK’S RUNTIME ORDERED BY TASKFLOW IN IICS\_EDW\_LOAD\_DETAILS\_CTL IN DEV**

SELECT MTT\_NAME,TASKFLOW\_NAME,DATEDIFF(second, MTT\_RUN\_START\_TIME, MTT\_RUN\_END\_TIME) AS DateDiff,

try\_to\_time(to\_char(DateDiff)) AS Time from "EDW\_DEV"."EDW\_LOAD\_CT"."IICS\_EDW\_LOAD\_DETAILS\_CTL"

ORDER BY

TASKFLOW\_NAME

**QUERY TO IICS MAPPING TASK’S RUNTIME ORDERED BY TASKFLOW IN IICS\_EDW\_LOAD\_DETAILS\_CTL IN DEV**

SELECT MTT\_NAME,TASKFLOW\_NAME,DATEDIFF(second, MTT\_RUN\_START\_TIME, MTT\_RUN\_END\_TIME) AS DateDiff,

try\_to\_time(to\_char(DateDiff)) AS Time from "EDW\_QA"."EDW\_LOAD\_CT"."IICS\_EDW\_LOAD\_DETAILS\_CTL"

ORDER BY

TASKFLOW\_NAME