# Intel-IAH CRM

User Guide for Intel IAH (Cloudera Hadoop) ingestion ELT framework.

Contents

[Intel-IAH CRM 1](#_Toc424308066)

[Quick start 3](#_Toc424308067)

[Requirements 3](#_Toc424308068)

[Build 3](#_Toc424308069)

[Deploy 3](#_Toc424308070)

[Run 3](#_Toc424308071)

[Configuration for Salesforce 3](#_Toc424308072)

[Detailed guide 5](#_Toc424308073)

[Configuration 5](#_Toc424308074)

[iah\_config 5](#_Toc424308075)

[JDBC config files 6](#_Toc424308076)

[SFDC data / metadata extraction and storage 7](#_Toc424308077)

[SFDC object export (to local FS) 7](#_Toc424308078)

[SFDC metadata export (to files or MDDB) 7](#_Toc424308079)

[Metadata reads (from files or MDDB) 7](#_Toc424308080)

[Dataset ingestion to HDFS / Hive 7](#_Toc424308081)

[Dataset merge in HDFS / Hive to latest version 7](#_Toc424308082)

[Script inventory and function 7](#_Toc424308083)

[Metadata Component 9](#_Toc424308084)

[Configuration Properties 11](#_Toc424308085)

# Quick start

## Requirements

Running requires a working Hadoop instance. The code expect to be deployed on a gateway / application server with working Hadoop configuration.

## Build

See the README.md document at the top level of the project.

## Deploy

Assuming the project was built from source, recursively copy the entire source directory (including the build artifacts) to a target location.

## Run

All scripts are in src/main/scripts by default.

### Configuration for Salesforce

* Set up $HOME/.iah\_config (from src/main/scripts templates)
  + Salesforce url and login
  + sf\_url=https://test.salesforce.com/services/Soap/c/31.0  
     sf\_username=username  
     sf\_password=password  
     sf\_usertoken=usertoken
  + Optional http/https proxy settings (http\_proxy,http\_proxy\_port, and https)
  + http\_proxy\_host=proxy-us.intel.com (optional)
  + http\_proxy\_port=911 (optional)
  + IAH JDBC config properties settings

iah\_jdbc\_config=/path/to/jdbc/config

* + If using a Kerberised environment, if not, leave these blank

krb\_principal=user@AMR.CORP.INTEL.COM

krb\_keytab\_path=/path/to/keytab/user.keytab

kerberos\_enabled=1

* Set up $HOME/.iah\_jdbc\_config (from src/main/scripts templates)
  + Use the default SQLite DB configuration.
* If using a Kerberised environment, run the krb\_add\_principal.sh to generate a keytab file for your Kerberos principal. This is a one-time process that needs to run before anything else is run. Once the keytab has been generated, it need not be run again.
* Create iah metadata db
* create\_iah\_db.sh
* Create SF objects in Hive / Impala

create\_hive\_dbs.sh

create\_sf\_objects.sh

* Run the extract, load, transform daily wrapper
* sf\_daily\_run.sh

# Detailed guide

## Configuration

### iah\_config

All configuration is specified by the iah\_config properties file. Settings in this file are respected / required by all scripts (and java programs they call).

The configuration file path defaults to ~/.iah\_config. The path can be overridden by the environment variable IAH\_CONFIG.

The properties file can be copied from the template in src/main/iah\_config.txt.

If multiple instances of the software will be operated on once host, they should each have their own iah\_config, with unique values for important properties listed below.

#### Important properties

These properties should be unique to a running instance and user.

##### hive\_schema

The default Hive DB for objects. Defaults to user\_$USER.

##### local\_root

The root for operations on the local filesystem. Defaults to /$HOME/iah

##### hdfs\_root

The root for data ingested into HDFS. Defaults to /user/$USER/data

The above properties are modified by data source specifications, stage of the process (ingest, snapshot, latest), job id / timestamp, etc., to determine actual paths use by data, DB, table and partition names.

##### iah\_jdbc\_config

Path to IAH DB connection properties file. See next section for contents of file.

#### Optional properties

, and any arguments such as a beeline connection properties file.

Note that beeline is required for secure (HiveServer2 + Sentry) clusters, but due to known bugs is disabled by default.

##### use\_schema\_inferrence

Use schema inference for text files instead of those found in metadata. Defaults to 0 (false).

##### default\_storage\_format

Storage format for objects created in Hive / Impala. Defaults to parquet.

##### sf\_username, sf\_password, sf\_usertoken, sf\_url

Salesforce configuration parameters.

##### user\_success\_list, user\_failure\_list

Mail notification lists for success, failure messages.

##### use\_single\_hive\_schema

Use a single hive schema for each stage instead of one per data source and stage

##### <data\_src>\_jdbc\_config, <data\_src>\_objects\_file

For JDBC data sources, path to JDBC config for that source and list of objects to extract. JDBC config uses sqlline compatible properties file. These follow the

### JDBC config files

All JDBC data source connection information is maintained in sqlline compatible properties files. These properties files define 4 keys:

driver – the JDBC driver class. Can be left blank for JDBC 4.X compatible drivers

url – the JDBC URL for the target data source

* + note that sqlite DB paths can be specified using forward slashes, even on Windows / Cygwin hosts.

user – user name, if required (sqlite doesn’t require this)

password – password, if required (sqlite doesn’t require this)

All 4 keys must be set, even if empty (url, driver, user, password).

#### iah\_jdbc\_config

This is a sqlline compatible properties file specifying the connection parameters for the metadata database.

The properties file can be copied from the template in src/main/iah\_jdbc\_config.txt. The path to this properties file is controlled by the iah\_jdbc\_config property in the main iah\_config properties file.

#### hive\_jdbc\_config, impala\_jdbc\_config

When set, these sqlline compatible properties files are used to define JDBC connection endpoints details for the HiveServer2 and Impala services.

These must be configured when using Sentry security.

## SFDC data / metadata extraction and storage

SFDC SOAP API client operations exposed via main.java.metadata.api.SFClientCmd class. All scripts to get data or metadata work through this class.

All scripts use SFDC configuration (login, pw, token, endpoint) from the master config file.

### Wrapper

# extract, ingest, merge and publish latest SFDC data

$ sf\_daily\_run.sh

### SFDC object export (to local FS)

# extract objects in sf\_objects\_file from SF endpoint to local working dir

$ dataloader/bin/extract\_sfdc.sh

### SFDC metadata export (to files or MDDB)

$ extract\_sf\_schemas.sh # to schema files, wraps SFClientCmd describe  
$ put\_sf\_schemas.sh # to MDDB, wraps SFClientCmd describe, put

## Eloqua data / metadata extraction and ingestion

### Wrapper

# extract, ingest, merge and publish latest Eloqua data

$ eloqua\_daily\_run.sh

### Extract

# extract Eloqua data

$ eloqua\_extract.sh

## JDBC data / metadata extraction and ingestion

### Wrapper

# extract objects in <data src>\_objects\_file from <data src>\_jdbc\_config

# load results into ingest and link to latest schemas

# schemas will be <hive\_schema>\_<data src>\_ingest and latest

$ jdbc\_copy\_run.sh <data src>

### Extraction only

# extract data from object to local FS, writing Avro schema and data file

$ extract\_jdbc.sh <cmd> <props> <qfile|table> <out file>

Note that if the query file argument is used, it can specify any select statement. The result table will be named for the basename of the query file (/path/to/query/account -> account).

### Query only

# Run sql cmd provided on STDIN against JDBC data source

$ extract\_jdbc.sh <props>

## Metadata writes (to MDDB)

$ put\_md\_schema.sh # save Avro schema to MDDB

## Metadata reads (from files or MDDB)

$ get\_md\_column\_names.sh <src> <table> # from MDDB  
$ get\_md\_table\_names.sh <src> # from MDDB

## Dataset ingestion to HDFS / Hive

Load all provided data into HDFS, create Hive metastore objects over it:

$ load\_all.sh <path to tablelist or directory> <src> <job\_id>

## Dataset merge in HDFS / Hive to latest version

Merge all tables from two ingest operations to resulting Hive table:

$ merge\_all.sh <path to tablelist> <src> <job\_id 1> <job\_id 2>

## Dataset publishing in HDFS / Hive to latest version

Link object views in latest schema to latest partition of corresponding object in snapshot schema:

$ create\_latest\_views.sh <path to tablelist> <src> <job\_id>

# Script inventory and function

src/main/scripts

* clear\_iah\_next\_run\_type.sh – forces next SFDC extraction to be full
* create\_analysis\_views.sh – constructs SFDC analysis views
* create\_hive\_dbs.sh – create needed Hive databases (or just their SQL)
* create\_iah\_db.sh - creates a default/empty metadata DB (MDDB)
* create\_jobid\_history.sh – create jobid history object
* create\_latest\_views.sh – create object views in latest schema
* create\_sf\_objects.sh – constructs a default/empty SFDC table object schema and analysis views
* disable\_iah.sh – disable iah from running via master wrappers
* enable\_iah.sh – enable iah to run from master wrappers
* extract\_all\_sf\_tables.sh - extracts all SFDC objects (deprecated, use dataloader scripts)
* extract\_jdbc.sh – extract data / schema from JDBC data source
* extract\_sf\_schemas.sh - extract all SFDC schemas to *files*
* extract\_sf\_table.sh - extract one SFDC table (deprecated, use dataloader scripts)
* get\_md\_column\_names.sh - print column names from MDDB for given source / table
* get\_md\_table\_names.sh - print list of tables from MDDB for given source
* get\_tab\_cols.pl - print list of column names for given hive table
* iah\_beeline.pl – wrapper for beeline that uses sqlline properties file
* iah\_config.txt - template for configuration file
* iah\_crm\_user\_data.sh – constructs a default user/master data table
* iah\_functions.sh - core script function library
* iah\_sqoop.pl – wrapper for sqoop that uses JDBC DSN file
* iah\_task\_runner.sh
* jdbc\_copy\_run.sh – wrapper for extract, load, transform from JDBC
* listall\_sf\_objects.sh – list all objects available from SFDC API
* kite-dataset - kite dataset CLI, used for csv imports
* load\_all.sh - ingestion wrapper to load many files at once into Hive
* log\_alert.sh – log alert to alert log system
* log\_task.sh – log task info to task log system
* materialize\_views.sh – materialize all views for data source
* merge\_all.sh - merge wrapper to merge multiple ingested datasets into latest
* merge\_table.sh - single table merge for testing
* migrate\_mviews.sh – migrate views from tables to table partitions
* order\_lovs.pl – load list of value data into Impala
* parse\_sfdc\_schema\_file.pl - parse SFDC schema file generated by extract\_sf\_schemas.sh.
* process\_lovs.sh – export list of values from MDDB, load into Impala
* put\_md\_schema.sh – put Avro schema into MDDB
* put\_sf\_schemas.sh - extract all SFDC schemas to MDDB
* run\_java.sh – run Java program with correct classpath, proxy, etc.
* run\_jdbc\_sql.sh – run sql commands against given JDBC DSN file
* run\_sqoop.sh – wrapper for iah\_sqoop.pl
* search\_task.sh – search task logs for latest task success or failure
* set\_iah\_next\_run\_type.sh – set the next iah run type to full or incr
* sf\_daily\_run.sh – wrapper for extract, load, transform from SFDC
* snapshot\_views.sh - deprecated
* sqlite\_exp\_mssql.sh – export SQLite DB for import into SQL Server
* sqlite\_exp.pl – transform SQLite SQL to SQL Server
* write\_md\_avro\_schema.sh – write an object from MDDB to Avro schema file
* write\_md\_hive\_schema.sh – write an object from MDDB to Hive schema file

bin

* compare\_sfdc\_schemas.sh – diff two trees of SFDC schema files
* eloqua\_all\_meta.py
* eloqua\_cdo\_meta.py
* eloqua\_daily\_run.sh – wrapper for extract, load, transform from Eloqua
* EloquaDeployment.sh
* eloqua\_extract.py
* eloqua\_extract.sh
* eloqua\_ingest\_ac.sh
* eloqua\_ingest\_all\_act.sh
* eloqua\_ingest\_cdo.sh
* eloqua\_ingest\_cn.sh
* eloqua\_ingest\_rest.sh
* eloqua\_rest.py
* excel\_to\_csv.py – convert Excel file to csv file (strips newlines)
* iah\_build\_deploy.sh – build and deployment script
* iah\_crm\_eloqua\_master.sh – Autosys wrapper for Eloqua process
* iah\_crm\_jdbc\_copy\_master.sh - Autosys wrapper for JDBC processes
* iah\_crm\_sfdc\_master.sh - Autosys wrapper for SFDC process
* iah\_crm\_user\_data\_db.sh – create and populate user\_data schema with local data
* view\_counts.sh – run view dashboard query against views for given job id
* smb\_extract.py – extract data from SMB share

dataloader/bin

* extract\_sfdc.sh – extract all objects (or from config list) from defined SFDC endpoint

# Metadata Component

The Metadata Component versions incremental changes in data source table objects, and enables schema version tracking and schema comparison by version numbers.

called the MetaData DataBase (MDDB)

Tools: a Sqlite/SQL Server database will be used to store all information in reference to: schema, data lineage, and logging for the IAH ETL processes.

1. Table Object: Base Table Object
2. Schema Object: Date based schema reference for the table object
3. Schema Column: Referenced columns in the Schema Objects
4. Column Type: Schema Column types
5. File Object: Path to file, file format, delimiter
6. TaskLog: Task-level identification, state
7. AlertLog: Alert identification, state

This component exposes APIs on 3 levels, through bash scripts and Java commands, and the Java component. The script API is dependent on the Java APIs, and the Java commands are dependent on the Java component.

These APIs enable table objects to be stored to and schema objects to be retrieved from the Metadata Component using the following method signatures.

1. Script API:
   1. put tables into the Metadata Component
   * put\_sf\_schemas.sh <tablename> <filepath> <timestamp> <username> <userpassword> <usertoken>
   1. get a list of table names
   * put\_md\_table\_names.sh <datasource> <timestamp>
   * get\_md\_column\_names.sh <datasource> <tablename> <timestamp>
   1. log information for a given task
   * log\_task.sh <job id> <task name> <object name> <cmd line> <env path> <event type> <event time> <log path> <host> <status> <rows processed>
   1. log information for a given alert
   * log\_alert.sh <job id> <task name> <object name> <cmd line> <env path> <event type> <event time> <log path> <host> <status> <rows processed> <desc> <action>
2. Java Command API:
   1. put tables into the Metadata Component
   * SFClientCmd putSchema <tablename> <filepath> <username> <userpassword> <usertoken>
   1. get a list of table names
   * MDClientCmd listables <datasource> <timestamp>
   * MDClientCmd getcolumns <datasource> <tablename> <timestamp>
3. put information for a given task
   * LClientCmd puttaskentry <task id> <task name> <object name> <cmd line> <env path> <event type> <event time> <log path> <host> <status> <rows processed>
   1. get information for a given task
   * LClientCmd gettaskentry <task id>
   1. put information for a given alert
   * LClientCmd putalertentry <task id> <task name> <object name> <cmd line> <env path> <event type> <event time> <log path> <host> <status> <rows processed> <desc> <action>
   1. get information for a given alert
   * LClientCmd getalertentry <task id>
4. Java Component API:
   1. put a table object into the Metadata Component
   * putSchema(String dataSource, String objectType, String ts, String filePath, String fileType, char delimiter, SchemaTable schema)
   1. get a list of table object names from the Metadata Component
   * SortedSet<String> getTableNames(String \_dataSource, String \_ts)
   1. get a list of columns for a given table object from the Metadata Component
   * SchemaTable getSchema(String dataSource, String objectType, String ts)
   1. save information for a given task using the Log Service

long putTaskEntry(TaskLog entry)

* 1. get information for a given task using the Log Service

TaskLog getTaskEntry(String taskId)

* 1. save information for a given alert using the Log Service

void putAlertEntry(AlertLog entry)

* 1. get information for a given akert using the Log Service
     + AlertLog getAlertEntry(String taskId)

# Configuration Properties

Complete list of configuration properties used by the software and how they are set. These properties are loaded via the load\_config method of the iah\_functions.sh library. All shell scripts in the IAH CRM project load their configuration using this method. Once loaded, they become available as variables in shell scripts.

|  |  |  |
| --- | --- | --- |
| Configuration property | Example | Notes |
| sf\_username | [iah@intel.com.crm.crmdev](mailto:iah@intel.com.crm.crmdev) | Set in config |
| sf\_password | foo | Set in config |
| sf\_usertoken | bar | Set in config |
| sf\_url | <https://test.salesforce.com/services/Soap/c/31.0> | Set in config – used by SOAP API for metadata |
| sf\_endpoint | <https://test.salesforce.com> | Set in config. Used by Apex Data Loader. |
| http\_proxy\_host | proxy-us.intel.com | Set in config. Automatically set in environment as needed. |
| http\_proxy\_port | 911 | Set in config. Automatically set in environment as needed. |
| https\_proxy\_host | proxy-us.intel.com | Set in config. Automatically set in environment as needed. |
| https\_proxy\_port | 911 | Set in config. Automatically set in environment as needed. |
| hive | hive | Override in config. Command used to execute hive client commands. |
| sf\_table\_exclude\_list | “case” | Set in config. Space separated list of tables to ignore. Also see sf\_objects\_file. |
| local\_root | /applocaldata/iah\_crm/sys\_iahcrm/qa | Override in config. Defaults to user home directory. |
| iah\_jdbc\_config | /appbin/iah\_crm/sys\_iahcrm/conf/qa\_iah\_jdbc\_config | Set in config. Path to JDBC config file for SQLLine. |
| hdfs\_root | /appdata/iah\_crm/sys\_iahcrm/qa/data | Override in config. Defaults to user HDFS home directory. |
| iah\_datasource | sfdc | Set in config. Used to construct default paths under local\_root, hdfs\_root and hive\_schema. |
| default\_impala\_server | hddev1.intel.com:25003 | Override in config. Defaults to localhost:21000. Not used if Sentry in use. |
| sf\_objects\_file | dataloader/conf/allsfdcobjectlatest | Set in config. Limits SFDC extraction to list of objects. |
| hive\_schema | iah\_crm\_qa | Override in config. Prefix used for all IAH CRM Hive database names. Defaults to user name. |
| hive\_dim\_schema | iah\_crm | Schema for dimension tables, which may be cached copies |
| latest\_views\_use\_real\_types | 0 | Override in config. Determines if views in \_latest DB use Hive casts from string to native SFDC types. |
| iah\_success\_msg | conf/iah\_success\_msg.txt | Set in config. Message template to parse and email on daily run success. |
| iah\_failure\_msg | conf/iah\_failure\_msg.txt | Set in config. Message template to parse and email on daily run failure. |
| iah\_success\_list | [user@intel.com](mailto:user@intel.com) | Set in config. Comma separated list of email addresses to notify on success. |
| iah\_failure\_list | [user@intel.com](mailto:user@intel.com) | Set in config. Comma separated list of email addresses to notify on failure. |
| scriptdir |  | Auto set. Refers to script directory src/main/scripts. |
| approot |  | Auto set. Refers to top level of project directory. |
| bindir |  | Auto set. Refers to bin directory. |
| confdir |  | Auto set. Refers to config directory. |
| libdir |  | Auto set. Refers to lib directory. |
| targetdir |  | Auto set. Refers to target directory. |
| ddldir |  | Auto set. Refers to ddl directory root. |
| dataloaderdir |  | Auto set. Refers to dataloader directory. |
| default\_umask | 0002 | Override in config. Defaults to 0002. |
| dataset | kite-dataset | Override in config. |
| kite\_max\_localfile\_size | 10000000 | Override in config. Max size of file before Kite uses MapReduce mode. |
| default\_hive\_server | localhost:10000 | Override in config. Endpoint for HiveServer 2. Not used if Sentry in use. |
| impala\_shell | impala-shell | Override in config. Command used to run Impala client commands. Not used if Sentry enabled. |
| iah\_tz\_offset | -0800 | Override in config. Used by Data Loader to set last run dates in expected format. |
| ingest\_partition\_key | ingest\_ts | Override in config. Name of partition key column for objects in ingest schema. |
| snapshot\_partition\_key | snapshot\_ts | Override in config. Name of partition key column for objects in snapshot schema. |
| ingest\_pk\_col | Id | Override in config. |
| snapshot\_pk\_col | Id | Override in config. |
| ingest\_mdate\_col | systemmodstamp | Override in config. |
| snapshot\_mdate\_col | systemmodstamp | Override in config. |
| ingest\_cdate\_col | createddate | Override in config. |
| snapshot\_cdate\_col | createddate | Override in config. |
| hdfs\_src\_root | $hdfs\_root/$iah\_datasource | Auto set via set\_data\_src call |
| hive\_dblocation |  | Override default location of Hive DBs. |
| hive\_secure | 0 | Override in config. If Hive and Impala are running Sentry. |
| log\_sql\_only | 0 | Override in config. If hive and impala clients should log SQL, but not execute it. Also useful for getting SQL and batching up for later execution |
| use\_md\_schemas | 0 | Override in config. If metadata service should be used where possible to describe data source object schemas into Hive / Avro schemas. |
| sfdc\_view\_list | $ddldir/sfdc/sqllist.txt | Override in config. List of views to create via create\_sf\_analysis\_views.sh. List should contain file names relative to the sqllist.txt path. |
| mat\_view\_list | $ddldir/sfdc/mat\_list.txt | Override in config. List of views to materialize via materialize\_views.sh. List should be view names in analysis schema. |
| default\_storage\_format | parquet | Override in config. default format for newly created (non-Avro) Hive / Impala objects |
| default\_objtype | view | default format for objects where materialization is optional. |
| local\_working\_dir | $local\_root/working/$ts | Auto set. Where temp files, etc. are put on local filesystem. |
| ts | Current time in YYYYMMDDHHMMSS | Auto set. Used to uniquely name batch runs, extracts, directories, and partitions. |
| sfdc\_schema\_dir | $local\_root/sfdc\_schema | Auto set. Where schemas are extracted to file system. |
| sfdc\_data\_dir | $local\_root/sfdc\_data | Auto set. Where data are extracted to file system. |
| ctldir | $local\_root/ctl | Auto set. Where process control files are stored. |
| iah\_control\_file | $ctldir/process.ctl | Auto set. Actual path to main process control file. |
| iah\_hostname | $HOSTNAME | Override. Defaults to current host name, used to uniquely name some files. |
| iah\_java\_home | /path/to/java/home | Override in config. Defaults to path returned by bigtop java detection. |
| iah\_java\_classpath | /appbin/path/to/jars | Additional Java classpath, used with all Java clients |
| driver\_libdir | /etc/sqoopjars | Additional Java classpath, only used with JDBC calls |
| hadoop\_heapsize | 2000 | Override in config. Passed on to Hadoop processes that respect HADOOP\_HEAPSIZE env variable. |
| hive\_schema\_eloqua | user\_$USER\_eloqua | Set in config. Root of schema for Eloqua tables. Should be deprecated. |
| eloqua\_site | IntelCorporation | Set in config. Eloqua REST API site name |
| eloqua\_username |  | Set in config. Eloqua REST API user name. |
| eloqua\_password |  | Set in config. Eloqua REST API password. |
| eloqua\_endpoint\_discovery\_url |  | Set in config. Eloqua REST API URL. |
| eloqua\_api\_version |  | Set in config. Eloqua REST API version. |