Integration Agent User Guide

Copyright 2018 Toolsverse LLC. All rights reserved.

Copyright 2018 Toolsverse LLC. All rights reserved.

What is Integration Agent

Integration Agent is a zero-maintenance, easy-to-configure, fully autonomous background service which runs behind the company's firewall and integrates various heterogeneous systems.

Usage Scenarios

Integration Agent supports the following usage scenarios:

1. Scheduling and executing ETL scenarios created in the Toolsverse Data Explorer.
2. Change replication from any database to any database. This includes automatic creation of the schema objects such as tables and indexes in the destination database.
3. Change replication from any database to the Snowflake data warehouse. This includes automatic creation of the tables in the Snowflake database.
4. Incremental backup any database to the CSV, JSON, XML and Excel files and copy them to the one of the following destinations: local file system, FTP, SFTP, Amazon S3.
5. Read CSV, JSON, XML and Excel files from the local file system, FTP, SFTP, Amazon S3 and load them into any database.

Basic Ideas

Integration Agent is installed as a Windows or Linux service and does not require any maintenance. It can track configuration changes on the fly and reconfigure itself without restarting.

Integration Agent can extract data from databases, files, web services and social networks. Agent can be configured to extract only the modified data, therefore supporting the change replication scenarios.

Copyright 2018 Toolsverse LLC. All rights reserved.

Extracted data can be loaded into the various destinations including databases, data warehouses such as Snowflake, files, web services, social networks, etc.

The streaming technology combined with the dataset partitioning allows Agent to migrate practically unlimited number of records even in the memory constraint environment.

Integration Agent emits heartbeat and monitoring events. To receive events, designated database must have tables in the format supported by Integration Agent.

Integration Agent simultaneously supports two modes: scheduling and executing XML scenarios created in Data Explorer and scheduling and executing property-driven scenarios.

XML scenarios

XML scenarios can be created in the Toolsverse Data Explorer and exported together with the required connections into the single XML configuration file.

There can be multiple configuration files. One file per XML scenario.

Configuration files can be scheduled using cron-based patterns or intervals in milliseconds. Each scheduled configuration is executed in its own thread.

Property-driven scenarios

The property-driven scenarios are configure using a single key-value configuration file.

Unlike XML scenarios, property-driven scenarios support only the following use cases: data migration with change replication and incremental data backup.

Datasets to extract data from can be combined in scenarios. Each scenario can have multiple datasets.

Each dataset can be loaded into the multiple destinations. By default, each dataset is extracted in its own thread.

Datasets can be partitioned based on the number of records. By default, each partition is loaded into the destination in its own thread.

Property-driven scenarios are scheduled using cron-based patterns or intervals in milliseconds. Each scheduled scenario is executed in its own thread.

Installing Agent and Configuring Agent as a Service

Installing Integration Agent

1. Download Integration Agent from the https://www.toolsverse.com/downloads/productdownloads.html
2. When downloading, select the appropriate version:
    a. Zip archive for the 64-bit versions of Windows. The download is bundled with the latest 64-bit Java JRE.
    b. Zip archive for the 32-bit versions of Windows. The download is bundled with the latest 32-bit Java JRE.
    c. Zip archive for all other operating systems, including all versions of Linux.

Copyright 2018 Toolsverse LLC. All rights reserved.

   d. Zip archive which contains the latest update: integrationagent-update-version-revision.zip.
3. Create folder for the agent in the hard drive, for example c:\agent or /usr/local/agent
4. Extract zip into this folder.
5. Copy provided license (the file with the extension **lic**) into the AGENT_HOME/license folder.

Windows

Installing Integration Agent as a Windows Service

1. Open elevated Command Prompt (Admin).
2. cd into the directory where you have installed the agent, for example c:\agent.
3. Execute `install.cmd`
4. Verify that service is installed by opening the Services desktop app. There must be a service with a name “IntegrationAgent” and description “Toolsverse LLC. Integration Agent”.

Removing Windows Service

1. Open elevated Command Prompt (Admin).
2. cd into the directory where you have installed the agent, for example c:\agent.
3. Execute `remove.cmd`
4. Verify that service is no longer in the list of the available Windows services.

Starting and Stopping Windows Service

Using Command Line

To **start**:

1. Open elevated Command Prompt (Admin).
2. cd into the directory where you installed agent, for example c:\agent.
3. Execute `start.cmd`

To **stop**:

1. Open elevated Command Prompt (Admin).
2. cd into the directory where you installed agent, for example c:\agent.
3. Execute `stop.cmd`

Using Windows Services Desktop App

To **start:**

1. Open Services desktop app.

Copyright 2018 Toolsverse LLC. All rights reserved.

2. Locate IntegrationAgent.
3. Click on service name. Click Start.

To **stop:**

1. Open Services desktop app.
2. Locate IntegrationAgent.
3. Click on service name. Click Stop.

Changing JRE and JVM memory parameters

The Agent (Windows versions) is bundled with the latest JRE (Java Runtime Environment).

JRE and JVM memory parameters are configured in **AGENT_HOME/install.cmd**.

```
set JAVA_HOME=%PR_PATH%\jre

set PR_JVM=%JAVA_HOME%\bin\server\jvm.dll

set PR_JVMOPTIONS=-Xmx1000m
```

You can change JRE to any valid JRE installed in the machine. You can also change the maximum memory that JVM can allocate. The default is 1000 Mb.

Important. You will need to remove service and install service again in order for any of these parameters to take effect.

Linux

Important. Make sure that the **Java 8** is installed on Linux. To verify run java -version from the terminal.

Java 10 and Java 11 are not currently supported.

Please note that you must have a root access to install the Agent as a Linux service.

Installing Integration Agent as a Linux service

Amazon Linux on AWS (CentOS)

1. Install jsvc.

```
sudo yum install jsvc
```

2. Copy provided **integrationagent** file into the **/etc/init.d/** folder
3. If needed modify **/etc/init.d/integrationagent** file using commands specific for your Linux distro. To edit file you can use the following command:

   ```
   sudo sed -i -e 's/\r//g' /etc/init.d/integrationagent
   ```

   For example, you might need to modify the JAVA_HOME variable or JVM parameters.

Copyright 2018 Toolsverse LLC. All rights reserved.

4. Make sure that **integrationagent** is executable
   `chmod +x /etc/init.d/integrationagent`

Ubuntu

1. Install jsvc.

```
sudo apt-get install jsvc
```

2. Copy provided **integrationagent** file into the **/etc/init.d/** folder
3. Modify **/etc/init.d/integrationagent:**

```
## Source function library for Amazon Linux.
#. /etc/rc.d/init.d/functions <- comment out this line

## Source function library for Ubuntu Linux.
. /lib/lsb/init-functions <- uncomment this line
```

You might also need to modify the JAVA_HOME variable or JVM parameters.

4. Make sure that **integrationagent** is executable
   `chmod +x /etc/init.d/integrationagent`

Starting and Stopping Linux Service

To **start:**

1. Open terminal
2. Type `sudo service integrationagent start`

To **stop:**

3. Open terminal
4. Type `sudo service integrationagent stop`

Changing JRE and JVM memory parameters

1. Open /etc/init.d/integrationagent in any editor.
   Example:

   ```
   sudo sed -i -e 's/\r//g' /etc/init.d/integrationagent
   ```

2. Modify JAVA_HOME and or JAVA_OPTS.
3. Save

Stopping Agent Using HTTP endpoint (Windows and Linux)

Copyright 2018 Toolsverse LLC. All rights reserved.

To **stop**:

1. Open any web browser
2. Enter http://web.server.host:web.server.port/stop

Example: http://localhost:8081/stop

Files and Folders

AGENT_HOME/config

Under this folder there is a main configuration file **integration-agent.properties**. This folder also contains the XML-based configurations, which include the XML scenarios and the connections.

AGENT_HOME/data

Under this folder Agent automatically creates folders **log** and **heartbeat** where it dumps current monitoring and heartbeat events as **csv** files. It moves them to the designated location and later to the monitoring tables in the database.

For each scenario agent creates a folder AGENT_HOME/data/**scenario_name**. When agent extracts data from the source it could create files in this folder. It then could move them to the designated destinations. Under this folder agent automatically creates **metrics** directory where it keeps latest metrics for the scenario in the metrics_**scenario_name**.csv file. Use this file to change a high-watermark used for the change replication.

AGENT_HOME/license

Integration Agent requires a valid, not expired license. License is a file with the extension **lic** stored in the AGENT_HOME/license folder.

Important. Integration Agent checks if existing license is still valid once a day at **01:01 local military** time. If license has expired while agent was still running it will log this and stop.

AGENT_HOME/logs

Integration Agent creates logs under this folder.

AGENT_HOME/jdbc

Integration Agent automatically loads JDBC drivers from this folder.

AGENT_HOME/lib

Integration Agent automatically loads Toolsverse and third-party libraries (jar files) from this folder.

AGENT_HOME/sql

SQLs for all major databases to create monitoring tables.

Copyright 2018 Toolsverse LLC. All rights reserved.

Upgrading agent to the new version

1. Stop Integration Agent if it is running.
2. Check current version of the agent by executing command

   `AGENT_HOME/integrationagent.exe` (Windows)

   or

   `java -jar integrationagent.jar` (Linux).

   If version is the same as or newer than the latest version in the Toolsverse website skip next steps – you already have the latest and greatest version of the agent. Otherwise proceed to the step 3.
3. Download integrationagent-**update**-version-revision.zip from the https://www.toolsverse.com/downloads/productdownloads.html.
4. Copy all files with subfolders from the zip to the Agent HOME, for example c:\agent. Answer "Yes" on all "File %filename already exists; do you want to override?" questions.
5. Start Integration Agent.

Command Line Tool

Integration Agent is designed to work as a background service. However, it can also be executed in a command line mode.

Execute Integration Agent in the standalone mode

Typically, Integration Agent is installed as a background service. You can also execute it as a standalone application.

- Windows - **`integrationagent.exe standalone`**
- All other operating systems - **`java -jar integrationagent.jar standalone`**

Display current version of the agent

- Windows - **`integrationagent.exe`**
- All other operating systems - **`java -jar integrationagent.jar`**

Additionally, when executed in the command line mode Integration Agent will check the license and display warning message if there is no valid license or license has expired.

Encrypt unencrypted passwords and passkeys

- Windows - **`integrationagent.exe encrypt property_name`**.

  `integrationagent.exe encrypt monitor.s3.password`

  You can substitute value of the property to encrypt on the value returned by agent:

  Executing `integrationagent.exe encrypt monitor.s3.password`

Copyright 2018 Toolsverse LLC. All rights reserved.

returns `3Des}7D35b8Pv+OG73tAIqZYDdqCVQLOfFeXa.`

You can then set `monitor.s3.password=3Des}7D35b8Pv+OG73tAIqZYDdqCVQLOfFeXa`

- All other operating systems - **`java -jar integrationagent.jar encrypt property_name`**

Configuring Integration Agent

To configure Integration Agent use **HOME/config/integration-agent.properties file**.

IMPORTANT: you can modify the property file without restarting the agent. It will automatically detect the changes and reconfigure itself.

Branding

Properties below are used for configuring branding:

```
app.vendor=Toolsevrse LLC.
app.title=Integration Agent
app.name=IntegrationAgent
```

Note. Property `app.name` additionally used as a file name when creating an application log.

Agent ID and Client Name

Each instance of the Integration Agent must have its own ID and the name of the client.

Note. Without properties below agent will not start:

```
agent-id=fa2f3f46-d522-11e8-9f8b-f2801f1b9fd1
client=Client Name
```

Monitoring configuration changes

When monitoring of the configuration is enabled the Integration Agent will automatically reconfigure itself if the configuration file has changed.

To monitor changes of the main configuration file **HOME/config/integration-agent.properties file** set the value of the property below to true:

```
watch.configuration.changes=true
```

Web Server

For remote monitoring purposes Integration Agent includes a built in Web server. Properties below are used for configuring web server's host and port. You can also disable the web server (it is enabled by default):

Copyright 2018 Toolsverse LLC. All rights reserved.

```
web.server.host=localhost

web.server.port=8081

# true is a default value

web.server.enabled=true
```

Controlling shutdown behavior

When Integration Agent if getting stopped or restarted, all currently running tasks must finish. **One of the following** events must occur:

1. There are no running scenarios and heartbeat tasks.
2. Waiting time until all tasks are finished has reached the configurable time in milliseconds.

```
# shutdown time out in milliseconds. Default is 300000 which is 5 minutes

shutdown.timeout=300000
```

Configuring Monitoring

This section is used to configure:

- Connection for the monitoring database.
- Heartbeat events and heartbeat scheduling.
- Monitoring events.
- What to do with the heartbeat and log files after they were uploaded into the monitoring database.
- S3 bucket used for storing monitoring events.
- The maximum number of threads used to copy monitoring files into the Amazon S3 bucket.
- The health rules.

Connection for the monitoring database (optional)

```
monitor.url= jdbc:oracle:thin:@localhost:1521:orcl

monitor.driver= oracle.jdbc.driver.OracleDriver

monitor.userid=user

monitor.password=password

monitor.params=additional parameters, for example internal_logon=sysdba

monitor.init.sql=sql to execute when creating connection
```

Enabling Recording Heartbeat events

```
# if this property is set to true (default is false) Integration Agent will
record heartbeat events in the monitoring database
```

Copyright 2018 Toolsverse LLC. All rights reserved.

```
monitor.heart.beat=true
```

Heartbeat scheduling

To configure how often heartbeat events will be emitted use property below. Valid cron-based patterns, as well as intervals in milliseconds are supported:

```
heartbeat.cron=*/5 * * * *
```

or

```
heartbeat.cron=60000
```

Note: The smallest interval when using a cron-based pattern is one minute. It is also possible to use milliseconds.

Last Heartbeat Event before Shutdown

When shutting down, the Agent emits the last heartbeat event with a status "**stopped**". It is possible to control how long the Agent will wait until the last heartbeat event will be uploaded into the monitoring database.

```
# heartbeat wait on shutdown time out in milliseconds. Default is 30000 which
is 30 seconds

heartbeat.on.shutdown.timeout=30000
```

Enabling Recording Metrics

```
# if this property is set to true (default is false) Integration Agent will
record metrics, such as the number of extracted and loaded records in the
monitoring database

monitor.report.metrics=true
```

Keep monitoring files

```
# default is false, meaning log and heartbeat files will be deleted

monitor.keep=false
```

S3 bucket to dump monitoring and heartbeat events (optional)

```
monitor.s3.url=bucket/incoming

monitor.s3.userid=s3 key id

monitor.s3.password=s3 key secret

monitor.s3.max.number.of.threads=10
```

Copyright 2018 Toolsverse LLC. All rights reserved.

```
monitor.s3.keep=true
```

Health Rules

When running, Integration Agent periodically performs a self-diagnostic and reports results as a part of the heartbeat event. It is possible to configure the health rules using properties below. The thresholds are configured in %.

```
memory.utilization.critical.threshold=90

memory.utilization.high.threshold=80

errors.ratio.critical.threshold=25

errors.ratio.high.threshold=10
```

Email Notifications

In this section, you can configure:

- The email sender.
- Who receives notification.
- What triggers notifications.

Email sender and who receives the notifications

```
mail.smtp.host=smtp.mail.company.com

mail.smtp.port=587

mail.smtp.user=

mail.smtp.password=

mail.smtp.to=user@toolsverse.com

mail.smtp.from=user@copmany.com

mail.smtp.cc=

mail.smtp.bcc=

mail.smtp.starttls.enable=true
```

What triggers the notifications

```
# send email notification on each successful data migration. Default is
false. Mail props must be set.
```

Copyright 2018 Toolsverse LLC. All rights reserved.

```
notification.onsuccess=false

# send email notification on each failed data migration. Default is true.
Mail props must be set.

notification.onfail=false

# send email notification on each failed log upload. Default is false. Mail
props must be set.

notification.onfail.log=false

# send email notification on each failed heartbeat upload. Default is false.
Mail props must be set.

notification.onfail.heartbeat=false

# send email notification on each shutdown. Default is false. Mail props must
be set.

notification.onshutdown=false

# send email notification on license expired event. Default is false. Mail
props must be set.

notification.onslicenseexpire=false
```

Configuring ETL logging

It is possible to configure logging for ETL scenarios.

```
# Global log level when executing ETL scenarios. Options:
info,warning,debug,severe,fatal. Default is severe

etl.log.level=severe


# Enabling separate log for each scenario. Options: false,true,unique.
Default is false. When value is set to unique the Agent will be creating a
unique log file for each scenario execution by adding a timestamp in
milliseconds from the epoch time to the log file name.

etl.log.each.scenario=false
```

Configuring XML scenarios

IMPORTANT: you can skip this section if you are only going to use property-driven scenarios.

Integration Agent can schedule and execute XML scenarios created in the Toolsverse Data Explorer. The scenarios are bundled with the connections and other settings in the single XML file and placed under the **AGENT_HOME/config** folder.

Copyright 2018 Toolsverse LLC. All rights reserved.

Schedule XML scenario

1. Create XML scenario in the Data Explorer.
2. Create XML configuration file.
3. Copy XML configuration file into the **AGENT_HOME/config** folder.
4. Add the following properties to the **AGENT_HOME/config/integration-agent.properties** file:

```
etl.config.scenario_name.cron=cron-based pattern or time in milliseconds
etl.config.scenario_name.file=name of the XML configuration file
```

Example

```
etl.config.test.cron=* * * * *
etl.config.test.file=etl_config.xml
```

Configuring property-driven scenarios

IMPORTANT: you can skip this section if you are only going to use XML scenarios.

Integration Agent can schedule and execute ETL scenarios driven by the properties in the **AGENT_HOME/config/integration-agent.properties** file.

Note: property-driven scenarios support only the following use cases: data migration with change replication and incremental data backup.

Main Snowflake Connection

Main Snowflake connection is required if the destination for the extracted data is a Snowflake data warehouse.

Note: Skip this section if you are not planning to load data into the Snowflake.

To configure Snowflake connection, use the following properties (example):

```
snowflake.url=jdbc:snowflake://company.snowflakecomputing.com:443
snowflake.driver=com.snowflake.client.jdbc.SnowflakeDriver
snowflake.userid=user
snowflake.password=password
```

Default Source Connection

This section is used to configure default connection to the database to extract data from.

Example:

```
source.driver=org.postgresql.Driver
source.url=jdbc:postgresql://localhost:5432
```

Copyright 2018 Toolsverse LLC. All rights reserved.

```
source.userid=postgres

source.password=password
```

Note. If you have multiple source connections, you can skip this section and configure the source connection for each scenario.

Default Extract Format

This section is used to configure the default file format for the extracted datasets. It is also used for configuring partitioning (setting the maximum number of records in a single file):

```
format.connector=com.toolsverse.etl.connector.text.TextConnector

format.connector.params=delimiter=,;charseparator=";metadata=false;firstrow=false;skipempty=true;suffix=timestamp;datetime=yyyy-MM-dd HH:mm:ss;date=yyyy-MM-dd;time=HH:mm:ss

# partitioning

format.max.rows.in.file=50000
```

Note. You can configure format for each scenario. Skip this section if you are configuring the change replication.

Sources

Typically, Integration Agent extracts data from the source database (will be explained later in the Scenario section).

Instead of pulling the data from the database, Integration Agent can also load the files directly from the file-based sources.

The following source types are supported:

- file - local file system
- ftp – FTP
- sftp – SFTP
- ftps – FTPS
- `com.toolsverse.io.CloudStorageProcessor` – S3

Important. When destination is a Snowflake, files must be in the format supported by the Snowflake COPY INTO command.

1. Create the file source (there are could be multiple). Example:

```
# source type
file.source.ftp_csv_source.transport=ftp
# IMPORTANT: include the full path in the format url/path/file. Wild cards for file name allowed
file.source.ftp_csv_source.url=ftp://ftp.toolsverse.com/test/employee*.csv
```

Copyright 2018 Toolsverse LLC. All rights reserved.

```
file.source.ftp_csv_source.userid=user
file.source.ftp_csv_source.password=password
# move is default, other option is copy
file.source.ftp_csv_source.action=move
# default is false. If value of this property is false files with the same
name + timestamp will be skipped
file.source.ftp_csv_source.allow.dups=false
```

2. Configure the dataset to get data from. Example:

```
scenario.load_files.EMPLOYEE.source=ftp_csv_source
scenario.load_files.EMPLOYEE.destinations=s3
# IMPORTANT: you can add post and pre SQL statements
```

3. Configure the COPY INTO SQL (for Snowflake destination only)

```
destination.s3.copy.to.snowflake.sql=COPY INTO "EMPLOYEE" FROM @STAGING
PATTERN = 'incoming/maxtest/EMPLOYEE_.*' FILE_FORMAT = (FORMAT_NAME =
CSVFORMAT) PURGE = TRUE
# IMPORTANT:
a) auto SQL is not supported,
b) If you are using wildcards for the files you must provide valid regex
c) make sure you are using a valid S3 folder
d) you must use the valid file format
e) if file was already copied into Snowflake it will not be uploaded again
so you must use unique file names
```

Destinations

Each dataset, either extracted from the database or pulled from the file, can be loaded into the one or multiple destinations.

Destinations can be shared between datasets. The following destination types are supported: database, local or network drive, FTP, SFTP, FTPS, S3.

Additionally, destination can be configured to require compressing files. The following compression algorithms are supported: **zip, gz**.

Destination can be configured to use multiple threads when copying files.

Format of the destination property

```
destination.name.property=value
```

Note. Name can be any word. For example: local, remote, home, favorite, etc.

Database Destination Example

```
destination.sql_server_test.driver=com.microsoft.sqlserver.jdbc.SQLServerDriv
er

destination.sql_server_test.url=jdbc:sqlserver://localhost:1433;DatabaseName=
test
```

Copyright 2018 Toolsverse LLC. All rights reserved.

```
destination.sql_server_test.userid=user
destination.sql_server_test.password=password
destination.sql_server_test.password=password
# default is false
destination.sql_server_test.auto.commit=true
# sql to execute when connection is created
destination.sql_server_test.init.sql=
```

File Destination Example

```
destination.local.transport=file
destination.local.url=/usr/local/myfiles
destination.local.folder=folder
destination.local.max.number.of.threads=10
```

FTP Destination Example

```
destination.ftp.transport=ftp
destination.ftp.url=ftp://ftp.company.com
destination.ftp.folder=test
destination.ftp.userid=user
destination.ftp.password=password
destination.ftp.zip.ext=zip
destination.ftp.max.number.of.threads=10
```

S3 Destination Example

```
destination.s3.transport=com.toolsverse.io.CloudStorageProcessor
destination.s3.url=bucket/folder
destination.s3.userid=key id
destination.s3.password=key secret
destination.s3.zip.ext=gz
destination.s3.copy.to.snowflake.sql=auto
destination.s3.max.number.of.threads=10
```

Copyright 2018 Toolsverse LLC. All rights reserved.

S3 Destination + Snowflake

S3 destination can be additionally configured to upload data into the Snowflake database using Main Snowflake Connection and configurable SQL.

Examples:

```
destination.s3.copy.to.snowflake.sql=auto
```

When SQL is set to auto, the agent will automatically generate the following SQL:

```
COPY INTO {TABLE} FROM @STAGING PATTERN = '{FOLDER}{TABLE}_.* FILE_FORMAT =
(FORMAT_NAME = CSVFORMAT) PURGE = TRUE
```

or

```
destination.s3.copy.to.snowflake.sql=actual SQL
```

In both cases token {FOLDER} will be substituted on S3 folder name (the string after the bucket name, for example bucket/**incoming**) and the token {TABLE} will be substituted on the table name. Please remember that TABLE=DATASET NAME (all capital). For example, if you named dataset "orders" there must be a table "ORDERS" in the Snowflake.

Scenario

Scenario combines one or more *datasets* to extract from, with each dataset linked to the one or more *destinations* to load data into.

Format for the scenario property

`scenario.name.property=value`

Note. Name can be any word. For example: high_volume_datasets, huge_extracts, etc.

Scheduling

To configure how often scenario must be executed use property `scenario.scenario_name.cron`. Valid cron-based patterns, as well as intervals in milliseconds are supported:

```
scenario.agent_test.cron=30000
```

or

```
scenario.agent_test.cron=* * * * *
```

Note: The smallest interval when using a cron-based pattern is one minute. It is also possible to use milliseconds.

Copyright 2018 Toolsverse LLC. All rights reserved.

Running Extracts in Parallel

By default, each dataset in the scenario is extracted in its own thread. You can use property below to disable parallel extracting:

```
scenario.agent_test.parallel=false
```

Disabling/Enabling copying files to the destination when dataset is empty

By default, empty datasets are not copied to the destination. You can enable it by using property

```
scenario.scenario_name.empty.datasets
```

Example

```
# false is a default value
scenario.agent_test.empty.datasets=true
```

ETL Drivers

By default, Integration Agent uses generic ETL drivers for the source and destination. If needed Agent can be configured to use database specific drivers.

Example:

```
scenario.agent_test.etldriver=com.toolsverse.etl.driver.sqlserver.MsSqlDriver

scenario.agent_test.sourceetldriver=com.toolsverse.etl.driver.sqlserver.MsSql
Driver
```

The following ETL drivers are supported:

- MS SQL Server - com.toolsverse.etl.driver.sqlserver.MsSqlDriver
- MySql - com.toolsverse.etl.driver.mysql.MySqlDriver
- PostgreSQL - com.toolsverse.etl.driver.postgres.PostgresDriver
- Oracle - com.toolsverse.etl.driver.oracle.OracleDriver
- DB2 - com.toolsverse.etl.driver.db2.Db2Driver
- Informix - com.toolsverse.etl.driver.informix.InformixDriver
- Sybase - com.toolsverse.etl.driver.sybase.SybaseDriver

Scenario specific Extract Format

By default, Integration Agent uses the Default Extract Format. You can configure format for the specific scenario using the properties below:

```
scenario.agent_test.format.connector=com.toolsverse.etl.connector.text.TextCo
nnector
```

Copyright 2018 Toolsverse LLC. All rights reserved.

```
scenario.agent_test.format.connector.params=delimiter=,;charseparator=";metad
ata=false;firstrow=false;skipempty=true;suffix=timestamp

scenario.agent_test.format.max.rows.in.file=10
```

Scenario specific Source Connection

By default, Integration Agent uses Default Source Connection. You can configure connection for the specific scenario using properties below (example):

```
# source database

scenario.agent_test.source.driver=com.microsoft.sqlserver.jdbc.SQLServerDrive
r

scenario.agent_test.source.url=jdbc:sqlserver://localhost:1433;DatabaseName=T
EST_DEV

scenario.agent_test.source.userid=user

scenario.agent_test.source.password=password

scenario.agent_test.source.readonly=true

scenario.agent_test.source.auto.commit=true

# sql to execute when source connection is created

scenario.agent_test.source.init.sql=
```

Datasets

Scenario must include one or more datasets to extract data from.

Format for the dataset property

```
scenario.name.dataset_name.property=value
```

Important. Dataset name can be any word but if the destination is a database and the table name to load data into is not specified (will be explained later) it **must** be the same as the database table name. Internally all dataset names are UPPERCASED. For example, if you named dataset “orders” there must be a table “ORDERS” in the database. It is possible to have destinations with a space in the name. Spaces must be encoded as \u0020 (UTF-8).

Example

```
NEW\u0020CFO\u0020SNAPSHOT is in fact “NEW CFO SNAPSHOT”
```

Copyright 2018 Toolsverse LLC. All rights reserved.

Destination table name

By default, Agent uses dataset name as a destination table name. You can override it by setting the table property.

Example:

```
scenario.agent_test.OFFICERS.table=Officers
```

Data Set Rank

Datasets are ordered by optional rank field. Ranking allows to the specify in which order datasets will be extracted - the lower rank the sooner dataset will be extracted and loaded into the destination(s).

Important. Ranking will not work if `scenario.scenario_name.parallel=true` (default).

INSERT/MERGE/UPDATE/DELETE

If destination is database, by default Agent generates "INSERT" SQL statements. In many cases it makes more sense to perform "MERGE" (UPSERT) instead. MERGE updates row if it already exists and inserts if it does not.

Example:

```
# possible actions: insert (default), merge, update and delete.

scenario.agent_test.AUDITEVENT.action=merge

# a comma separated list of the fields used to lookup the record

scenario.agent_test.AUDITEVENT.keys=AuditEventId
```

Full Extract and Delta Extract

When data is extracted from the databases, Agent can either do a full extract, a delta extract or both (conditionally).

- Full Extract - extract the whole dataset (table, select statement, etc.)
- Delta Extract – extract only the part of the dataset which has been changed since the last extract
- Both – do full extract first time, do subsequent delta extracts

SQL for Full Extract

To configure SQL for the full extract use the following property `scenario.name.dataset_name.full=sql`.

Example

```
scenario.agent_test.orders.full=select * from orders
```

Copyright 2018 Toolsverse LLC. All rights reserved.

High Watermark

Integration agent records and uses a high watermark value for delta extracts. There are two options:

1. The maximum value of the configurable field in the data set, calculated when running last successful extract. To configure which field to use for high watermark use property `scenario.scanario_name.dataset_name.high.watermark.field.`

   Example

   ```
   scenario.agent_test.ITEMS.high.watermark.field=last_updated
   ```

   Important. Any date+time or number field can be used as a high watermark.

2. The start system timestamp of the last successful extract. The timestamp is calculated automatically and there is nothing to configure. Disadvantage – if time is different on the box running Agent and the database server the option number 2 can cause duplicates or missing records.

Important. Option number 1 is preferable. It does not depend on time zones and time differences between box running Agent and the database server.

Resetting the high watermark

Each time agent successfully extracts the data from the databased and loads the data into the designated destinations it updates the file HOME/data/scenario_name/metrics/metrics_ scenario_name.csv. The file looks like below:

```
name,extractStarted,extractFinished,numberOfExtractedRecords,exceptionDuringExtract,loadStarted,loadFinish
ed,numberOfLoadedRecords,exceptionDuringLoad,lastSuccessfulExtractStarted,lastSuccessfulExtractFinished,la
stSuccessfulLoadStarted,lastSuccessfulLoadFinished,highWatermarkOnExtract,highWatermarkOnLoad

ITEMS,,,0,,2016-02-01 16:55:04,2016-02-01 16:55:07,7779,,,,2016-02-01 16:55:04,2016-02-01 16:55:07.123,,
2016-02-01 16:55:07.123456

ORDERS,,,0,,2016-02-01 16:55:01,2016-02-01 16:55:04,434343,,,,2016-02-01 16:55:01,2016-02-01
16:55:04.456,,2016-02-01 16:55:07.123456
```

The first highlighted in **bold** field is a number of loaded records, the second one is a timestamp of the last **successful** load and the last one is a **high watermark** of the last successful load. You can manually change the timestamp to, for example, empty string or any valid date+time in the format yyyy-MM-dd HH:mm:ss.SSS.

If the value of the high watermark field is extracted from the TIMESTAMP field it is recorded in the format yyyy-MM-dd HH:mm:ss.ffffff, where ffffff is a time in nanoseconds. Make sure to use the same format when manually setting the high watermark to the not null value.

SQL for Delta Extract

To configure SQL for the delta extract, use the following property:

`scenario.name.dataset_name.delta=sql`

Copyright 2018 Toolsverse LLC. All rights reserved.

Example

```
scenario.agent_test.orders.delta=select * from orders where last_updated >
{ORDERS_HIGH_WATERMARK }
```

Note. When configuring delta extracts the following tokens are available (ALL CAPITAL):

- {DATASET_NAME_HIGH_WATERMARK} – the calculated high watermark (option 1)
- {DATASET_NAME_LAST_TIMESTAMP} – timestamp of the last extract (option 2)
- {DATASET_NAME_LAST_DATE} – date of the last extract (option 2)
- {DATASET_NAME_LAST_TIME} – time of the last extract (option 2)

Execute any SQL on the destination connection before loading data

To execute any SQL before loading data into the destination database use the following property: `scenario.name.dataset_name.before.sql.`

Example

```
scenario.agent_test.orders.before.sql= delete from PURCHASEORDERHEADER
```

Note. You can configure agent to execute any number of ";" delimited SQL statements.

Important. If you want Agent to ignore any exceptions which occurred when executing before SQLs use property `scenario.agent_test.orders.before.sql.ignore.exception`

Example

```
scenario.agent_test.orders.before.sql.ignore.exception=delete from
PURCHASEORDERHEADER
```

Execute Any SQL on the destination connection after loading data

To execute any SQL after loading data into the destination database use the following property: `scenario.name.dataset_name.after.sql.`

Example

```
scenario.agent_test.orders.after.sql= merge into PURCHASEORDERHEADER
```

Note. You can configure agent to execute any number of ";" delimited SQL statements.

Important. If you want Agent to ignore any exceptions which occurred when executing after SQLs use property `scenario.agent_test.orders.after.sql.ignore.exception`.

Example

```
scenario.agent_test.orders.after.sql.ignore.exception=merge into PURCHASEORDERHEADER
```

Copyright 2018 Toolsverse LLC. All rights reserved.

Always force Full Extract

To always force the full extract, regardless of the high-watermark value use the property `scenario.name.dataset_name.always.full`

Example

```
scenario.agent_test.orders.always.full= true
```

Important. Default value is false. The full extract SQL must be configured.

Datasets with spaces in names

It is possible to create the datasets with spaces in the name. The spaces are encoded using the standard UTF-8 value for the space character: \u0020.

Example:

```
scenario.agent_test.NEW\u0020CFO\u0020SNAPSHOT.full=SELECT * FROM
CFO_SNAPSHOT
scenario.agent_test.NEW\u0020CFO\u0020SNAPSHOT.destinations=s3
```

Important. You don't need to do anything special if the dataset has columns with spaces. The Agent will automatically add double quotes around the table and field names with spaces.

Destinations

Each dataset can be loaded into the one or multiple destinations. Use coma (,) separated list of *destinations* as a value of the property `scenario.name.dataset_name.destinations.`

Example:

```
scenario.agent_test.orders.destinations=local,s3
```

Monitoring

You can monitor Integration Agent using one of the following methods.

Logs

Logs are created in the HOME/logs folder. By default, logs are rolled daily. The following logs are available:

- IntegrationAgent.log – main log file. Look for all errors here
- commons-daemon.log – service events such as staring, stopping, etc.
- integrationagent-stderr.log – std error output
- integrationagent-stdout.log – std output
- **scenario_name**-timestamp.log – per scenario logs if enabled

Copyright 2018 Toolsverse LLC. All rights reserved.

To fine tune the configuration of the logger use file **AGENT_HOME/config/lo4j.properties**.

Changing ETL log level

By default, the Agent is configured to log only errors when executing ETL scenarios. To change the log level used specifically when executing ETL scenarios modify these properties in the **HOME/config/integration-agent.properties** file.

Web Dashboard

To open a web dashboard:

1. Open any web browser
2. Enter http://*web.server.host*:*web.server.port*

Example: http://localhost:8081

Health Endpoint

To get health information in the JSON format:

1. Open any web browser (or use any tool which can make and process http requests, such as curl, Postman, etc.)
2. Enter http://web.server.host:web.server.port/health

Example: http://localhost:8081/health

Heartbeat and Monitoring events

Agent can be configured to emit perioding heartbeat events. It can also generate monitoring events each time the ETL scenario is executed.

The following destinations for the monitoring events can be configured:

- S3 bucket
- Database
- Snowflake database

Please use this section of documentation to configure the monitoring destination.

Monitoring tables in the database

To insert monitoring events into the database:

1. Execute HOME/sql/dbname_init.sql in the database used for monitoring.
2. Configure monitoring connection.
3. Enable recording monitoring and heartbeat events.
4. Schedule the heartbeat monitoring.

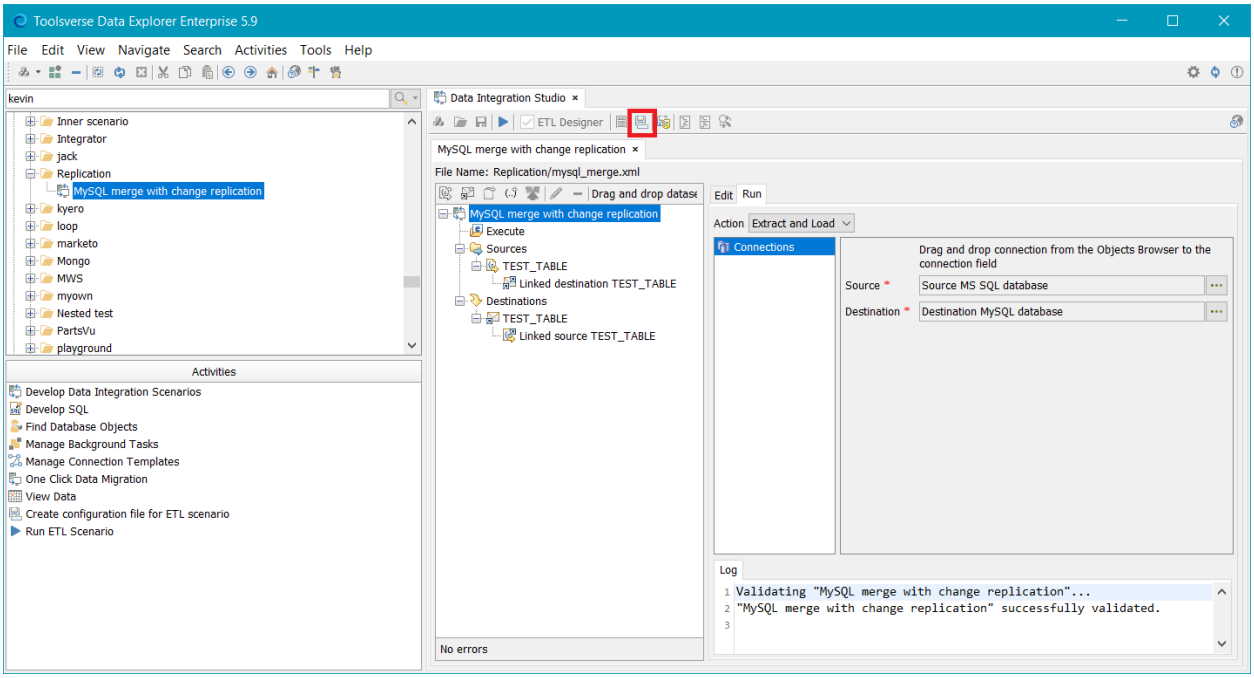
Copyright 2018 Toolsverse LLC. All rights reserved.

Use Cases

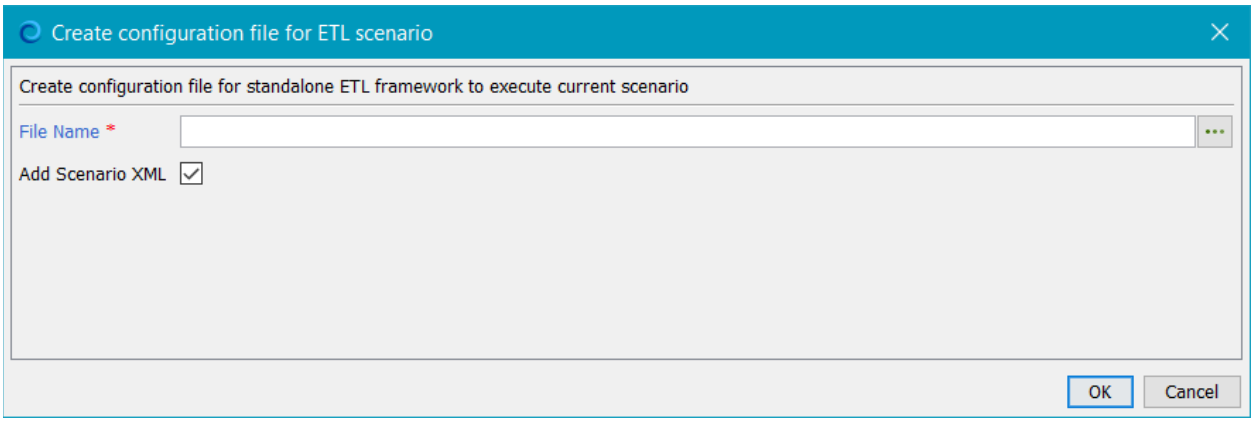
Creating and scheduling XML scenarios

1. Create XML scenario using Toolsverse Data Explorer.
2. Create XML configuration file which includes XML scenario and the required connections.

STEP 1: Click the button “Create Configuration File for ETL scenario”



STEP 2: Enter the file name. Keep “Add Scenario XML” selected.



3. Copy XML configuration file into the **AGENT_HOME/config** folder.
4. Schedule XML scenario.

Recommended settings for the XML scenarios

1. Heartbeat and monitoring events are enabled.
2. Heartbeat monitoring is scheduled.

Copyright 2018 Toolsverse LLC. All rights reserved.

3. Saving events into the database is enabled.

Setting up a brand-new full extract from the database (property-driven scenario)

Full extract implies that the entire dataset is getting extracted from the source and loaded into the destination.

Steps:

1. Add dataset to extract from. Add suffix **full** to the dataset name.

   Important. If the destination is a database table and it does not exist it will be automatically created based on the metadata returned by the select statement. All source data types will be mapped to the destination data types. The NOT NULL constraints will be preserved.

   Example:
   `scenario.agent_test.PURCHASEORDERHEADER.full=SELECT * FROM [PURCHASING].[PURCHASEORDERHEADER]`

2. Add destinations to copy data to (can be a comma separated list of destinations).

   Example:
   `scenario.agent_test.PURCHASEORDERHEADER.destinations=dest_database`

3. Specify destination table name.

   Example:
   `scenario.agent_test.PURCHASEORDERHEADER.table=Order`

4. Make sure that there is no row in the **AGENT_HOME/data/scenario_name/metrics/metrics_scenario_name.csv** file for this dataset. **Important.** If row exists – delete the whole row.

Setting up a brand-new delta extract from the database (property-driven scenario)

Delta extract is an incremental extract. Only changes since the last extract are getting loaded into the destination.

1. Add dataset to extract from. Add suffix **delta** to the dataset name.

   When configuring delta extracts the following tokens are available (ALL CAPITAL):

   - `{DATASET_NAME_HIGHT_WATERMARK} – high watermark of the last extract`
   - `{DATASET_NAME_LAST_TIMESTAMP} – timestamp of the last extract`
   - `{DATASET_NAME_LAST_DATE} – date of the last extract`
   - `{DATASET_NAME_LAST_TIME} – time of the last extract`

   Add other relevant conditions.

Copyright 2018 Toolsverse LLC. All rights reserved.

Important. If the destination is a database table and it does not exist it will be automatically created based on the metadata returned by the select statement. All source data types will be mapped to the destination data types. NOT NULL constraints will be preserved.

Example

```
scenario.agent_test.PURCHASEORDERHEADER.delta=SELECT * FROM
[PURCHASING].[PURCHASEORDERHEADER] where MODIFIEDDATE >
{PURCHASEORDERHEADER_HIGH_WATERMARK}
```

2. If you are using HIGH WATERMARK you will need so specify a high watermark field in the source table. The latest (greatest) value for this field will be used as a HIGH_WATERMARK.

   Example

   ```
   scenario.agent_test.PURCHASEORDERHEADER.high.watermark.field=ModifiedDate
   ```

3. Add destinations to copy data to (can be a comma separated list of the destinations). If the full extract was already configured – skip this step: full and delta extracts share the same destinations.

   Example:
   ```
   scenario.agent_test.PURCHASEORDERHEADER.destinations=test_destination
   ```

4. Make sure that there is no row in the HOME/data/scenario_name/metrics/metrics_scenario_name.csv file for this dataset.
   Important. If row exists – delete the whole row.

Configuring the Change Replication (property-driven scenario)

When change replication is configured Integration Agent tracks the changes in the source database and migrates only new and updated records into the destination database.

To configure change replication:

1. Configure full extract for the first extract.
2. Configure delta extract for subsequent extracts.
3. Configure merge if needed.
4. Specify the order of datasets to extract and load (optional).

Configuring Change Replication into the Snowflake (property-driven scenario)

Agent supports direct load into the Snowflake data warehouse. Loading data into the Snowflake requires configuring the Snowflake to be able to load data from the staging S3 bucket.

The simplified short version of the flow looks like below:

1. Data is getting extracted from the source database or directly from the CSV files.

Copyright 2018 Toolsverse LLC. All rights reserved.

2. Extract files are created and moved into the staging S3 bucket.
3. Snowflake COPY INTO command is executed.

To configure change replication into the Snowflake:

1. Configure Main Snowflake Connection.
2. Configure staging S3.
3. Configure full extract for the first extract.
4. Configure delta extract for subsequent extracts.
5. Specify the destination as S3

Example

```
scenario.agent_test.AUDITEVENT.destinations=s3
```

Configuring Incremental Database Backup (property-driven scenario)

Agent can be configured to extract data from the database, create files (zip if needed) and copy them into the designated location (file system, ftp, sftp, s3).

Steps:

1. Configure destination(s).
    a. File
    b. FTP
    c. SFTP
    d. S3
2. Configure full extract for first extract.
3. Configure delta extract for subsequent extracts.

Load files them into the database (property-driven scenario)

Follow this instruction to configure the file load.

Resetting the automatic full extract (property-driven scenario)

The automatic full extract is typically executed just one time. Once this dataset in added to the HOME/data/scenario_name/metrics/metrics_scenario_name.csv file and the value of the lastSuccessfulLoadStarted field is not null agents stops executing the full extract.

To reset set the value of **lastSuccessfulLoadStarted** and **highWatermarkOnLoad** fields to empty. You can also set it to any valid date+time in the yyyy-MM-dd HH:mm:ss.SSS (yyyy-MM-dd HH:mm:ffffff for high watermark) format.

Before

```
PURCHASEORDERHEADER,,,0,,2016-02-09 13:53:10,2016-02-09 13:53:10,0,,,,2016-02-09 13:53:10.123,2016-02-09 13:53:10.123,,2016-02-09 13:53:10.123456
```

Copyright 2018 Toolsverse LLC. All rights reserved.

After

```
PURCHASEORDERHEADER,,,0,,2016-02-09 13:53:10,2016-02-09 13:53:10,0,,,,,2016-02-09 13:53:10,,
```

Important. Once you reset the full extract, the Agent will perform the full extract again, so if the destination for this dataset is a database it is recommended to delete all records from the destination table or drop the table all together (in this case the Agent will automatically create it).

It is also possible to always execute full extract.

Resetting a delta extract (property-driven scenario)

Resetting a delta extract is performed similarly to resetting a full extract.

Important. After resetting the extract, the Agent will try to execute the full extract. If the full extract is not configured – nothing will be executed.

Executing multiple before and after SQL statements (property-driven scenario)

It is possible to execute multiple ";" separated SQL statements before and after the load. The Agent will parse the SQL, split it on multiple statements, prepare each statement separately, set bind variables and execute them one by one.

Example:

```
scenario.agent_test.orders.before.sql= delete from test; delete from abc;
```

Important. If you want to have each SQL in its own line use \ as an end-of-line character:

```
scenario.agent_test.orders.before.sql= delete from test;\
delete from abc;\
insert into abc values(…);
```

Performance Tuning

By default, the Agent is configured with a highest possible level of performance in mind. By changing properties, you can severely impact the performance (for better or worse).

Quick performance tuning tips:

1. **Important.** Make sure that all extract queries have required index support. If you are using any condition in the WHERE clause the table must have the corresponding indexes.
   Example: `SELECT * FROM [PURCHASING].[PURCHASEORDERDETAIL] WHERE MODIFIEDDATE > {PURCHASEORDERDETAIL_LAST_TIMESTAMP}`
   If there is no index for the field MODIFIEDDATE, executing query above on a large dataset will cause a huge performance problem. The database will have to perform a full table scan on the PURCHASEORDERDETAIL table.

Copyright 2018 Toolsverse LLC. All rights reserved.

2. If possible do not disable parallel extract (`scenario.scenario_name.parallel=true`)
3. Set maximum number of threads for destination based on the formula number_of_cpu_cores * 2: destination.ftp.max.number.of.threads=10
4. Keep emptydatasets property set to false (default): scenario.scenario_name.emptydatasets=false. It is set to false the empty datasets will not be copied to the destinations.
5. Use scheduling in milliseconds for testing purposes only. Use cron-based patterns in production.

Copyright 2018 Toolsverse LLC. All rights reserved.