Supported File Formats

Contents:

This section contains information on the fie formats and compression schemes that are supported for input to and output of Trifacta® Self-Managed Enterprise Edition.

NOTE: To work with formats that are proprietary to a desktop application, such as Microsoft Excel, you do not need the supporting application installed on your desktop.

Filenames

NOTE: Filenames that include special characters can cause problems during import or when publishing to a file-based datastore. Do not use the slash (/) character in your filenames.

Native Input File Formats

Trifacta® Self-Managed Enterprise Edition can read and import directly these file formats:

- Excel (XLS/XLSX)

  Tip: You may import multiple worksheets from a single workbook at one time. See *Import Excel Data*.

- CSV
- JSON, including nested

  NOTE: Trifacta Self-Managed Enterprise Edition requires that JSON files be submitted with one valid JSON object per line. Consistently malformed JSON objects or objects that overlap linebreaks might cause import to fail. See *Initial Parsing Steps*.

- Plain Text
- LOG
- TSV
- XML

  Tip: XML files can be ingested as unstructured text.

- Avro

NOTE: Trifacta Self-Managed Enterprise Edition supports Hive connectivity, which can be used to read data for Hadoop file formats that are not listed here, such as Parquet. For more information, please view the documentation for your Hive version.

For more information on data is handled initially, see *Initial Parsing Steps*.

Native Output File Formats

Trifacta Self-Managed Enterprise Edition can write to these file formats:

- CSV
- JSON

- Tableau (TDE)

  NOTE: Publication of results in TDE format may require additional configuration. See below.

- Avro

  NOTE: The Photon and Spark running environments apply Snappy compression to this format.

- Parquet

  NOTE: The Photon and Spark running environments apply Snappy compression to this format.

Compression Algorithms

NOTE: Importing a compressed file with a high compression ratio can overload the available memory for the application. In such cases, you can uncompress the file before uploading.
Or, if that fails, you should contact your administrator about increasing the Java Heap Size memory.

NOTE: Publication of results in Snappy format may require additional configuration. See below.

NOTE: GZIP files on Hadoop are not split across multiple nodes. As a result, jobs can crash when processing it through a single Hadoop task. This is a known issue with GZIP on Hadoop.

Where possible, limit the size of your GZIP files to 100 MB of less, or use BZIP2 as an alternative compression method. As a workaround, you can try to run the job on the unzipped file. You may also disable profiling for the job. See *Run Job Page*.

Read Native File Formats

	GZIP	BZIP	Snappy
CSV	Supported	Supported	Supported
JSON	Supported	Supported	Supported
Avro			Supported
Hive			Supported

Write Native File Formats

	GZIP	BZIP	Snappy
CSV	Supported	Supported	Supported
JSON	Supported	Supported	Supported
Avro			Supported; always on
Hive			Supported; always on

Additional Configuration for File Format Support

Publication of some formats requires execute permissions

When job results are generated and published in the following formats, the Trifacta platform includes a JAR, from which is extracted a binary executable into a temporary directory. From this directory, the binary is then executed to generate the results in the proper format. By default, this directory is set to `/tmp` on the Trifacta node.

In many environments, execute permissions are disabled on `/tmp` for security reasons. Use the steps below to specify the temporary directory where this binary can be moved and executed.

Steps:

1. Login to the application as an administrator.
2. From the menu, select **Settings menu > Settings > Admin Settings**.
3. For each of the following file formats, locate the listed parameter, where the related binary code can be executed:

File Format	Parameter	Setting to Add
Snappy	`"data-service.jvmOptions"`	`-Dorg.xerial.snappy.tempdir=<some executable directory>`
TDE	`"batch-job-runner.jvmOptions"`	`-Djna.tmpdir=<some executable directory>`

4. Save your changes and restart the platform.
5. Run a job configured for direct publication of the modified file format.