

Description: A general overview of Data Exporter, and how to configure it.

General Requirement: Basic knowledge of Hyper Historian and its architecture and basic knowledge of your desired storage (Microsoft SQL Server, Azure SQL, Azure Data Lake, Apache Kafka, Apache Hadoop, or local folder for CSV files).

Introduction

New in version 10.96, the Data Exporter is a powerful, high-performance extension module for Hyper Historian that makes it easier to regularly export your logged data to third party storage components such as Microsoft SQL Server Azure Data Lake, Apache Kafka, Apache Hadoop and others.

General Overview

Data Exporters work by scheduling tasks to sort logged data into a dataset and then transfer that dataset into a storage space, such as a data lake.

The data explorer currently supports these storage types:

- Microsoft Azure Data Lake
- Microsoft Azure SQL
- Microsoft SQL Server
- Apache Kafka
- Apache Hadoop
- Local CSV Files

Users who previously used the Hyper Reader – a command-line tool to export data from Hyper Historian into a SQL Server database – should be able to use data exporters to accomplish the same tasks without having to use the command line. The Hyper Reader is still included for backwards compatibility, but we expect users will prefer to use the Data Exporter and ICONICS recommends it for new projects.

Data exporters are configured in Workbench under **Historical Data > Hyper Historian > Data Exporters**:

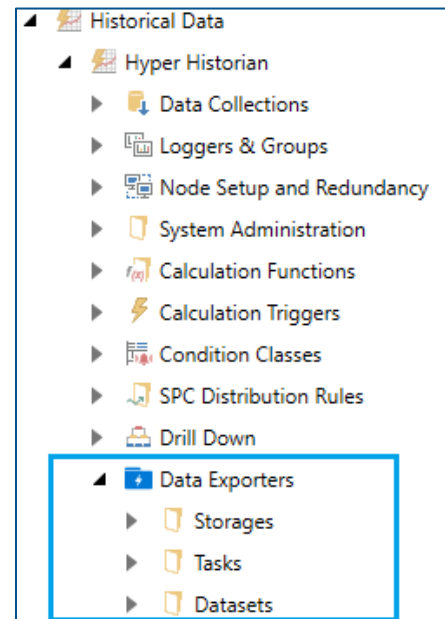


Figure 1 – Location of Data Exporters

The **Storage** folder is where you configure the connections to data storage, such as SQL Server, Azure SQL, Apache Kafka, Apache Hadoop and others.

The **Datasets** folder is where you configure the tags, filters, and the format (columns) of the data you want to export. The data can be metadata, raw, or aggregated values.

The **Tasks** folder is where you choose a storage and one or more datasets and configure how often you would like the data to be exported. Each time the task runs, it will export the data logged between this run and the last run.

Note: The first time a task runs it will not actually copy any data. It just marks the beginning of the range of data to copy. The

Synchronization Offset configured on a task can be used to copy slightly further back than the last task run. The actual start time of the copy is the last run minus the synchronization offset, and the end time is the present.

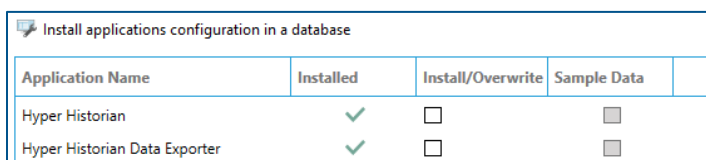
Enabling the Data Exporter

The Hyper Historian Data Exporter has its own set of configuration tables, which must be added to the same database as the Hyper Historian configuration.

To add the Data Exporter to an existing project, open **Workbench**, go to the **Project** tab, and select **Configure**

Database. Choose your unified configuration or Hyper Historian database.

On the **Install applications configuration in a database** page, check to see if the **Installed** column is checked for the **Hyper Historian Data Exporter**. If it is already checked, you can cancel out of this dialog. If it is not checked, check the **Install/Overwrite** box for the **Hyper Historian Data Exporter**, then click **Install**.



Application Name	Installed	Install/Overwrite	Sample Data
Hyper Historian	✓	☐	☐
Hyper Historian Data Exporter	✓	☐	☐

Figure 2 – Installing the Data Exporter tables

Once the configuration has been installed, right-click on your project and select **Configure Application(s) Settings**. Make sure that the Hyper Historian Data Exporter is enabled and configured to use the same database as Hyper Historian.

Lastly, you will need to make sure the extension is enabled under **Historical Data > Hyper Historian > System Administration > Installed Extensions**. If you do not plan to use the Data Exporter, you can disable it on this page to save system resources.

Example: Export to SQL Server Database

This example will walk you through configuring an export task to copy data into a SQL Server database.

Exporting to other storage types is very similar, simply choose the appropriate Connection Type when creating your Storage object and fill in the appropriate connection information. You can find sections later in this document that describe how to find the connection information for other storage types we support.

Storage Configuration

1. In **Workbench**, expand **Historical Data > Hyper Historian > Data Exporters**.
2. Right-click on **Storage** create a new data storage.
3. Name this storage, **Local SQL**.
4. For **Connection Type**, select **SQL / Azure SQL**.
5. Select the **Configure connection** hyperlink under **Connection String**.

6. Follow the dialog to connect to your SQL Server database. Unless you have a specific database in mind, we recommend creating a new database by typing a name into the **Database** field. (The credentials you provide must have the proper permission to create a database.)

Note: If you installed SQL Express with your ICONICS installation, that instance will be named **localhost\SQLEXPRESS2017**.

7. You should now have a connection string in the **Connection String** field.
8. If desired, adjust the settings in the **Advanced** section.
9. Click **Apply**.

Datasets Configuration

1. Right-click on **Datasets** and add a dataset definition.
2. Name this definition **SimulatedData**.
3. Go to the **Columns** tab.
4. If desired, add more columns or modify the existing columns.
5. Go to the **Filters** tab.
6. Click to add a new item in the **Filters** table.
7. On the new row, click in the **Path Filter** column and select the **folder** button.
8. Select the **Various Signals** folder. Select **OK**, then **Apply**. This will export data for all the tags in the Various Signals folder.

If desired you could further filter the tags in that folder by using the Point Name Filter column, or specific data types by using the Variable Type Filter.

The Variable Type Filter column accepts these values:

- R (for raw data)
- C (for calculated data)
- A (for aggregated data)

Note: If you have specific tags you would like to log, you can use the **Tags** table instead of **Filters**. If both the **Tags** and **Filters** are enabled, tags that meet either condition will be included. (It is an "or" relationship.)

This example was for raw data. If you would like to export aggregated data, make sure to select **Aggregated** for **Storage data** on the **Dataset definition** tab, then configure the **Aggregate Options**. To choose the actual aggregate, on the

Columns tab, add a column with a **Column Value Type** of **Value** and a **Resampled Value Type** of your desired aggregate.

Tasks Configuration

1. Right-click on **Tasks** and add a synchronization task.
2. Name this task **SignalsTask**.
3. Under **Storage**, select the **Local SQL** connection, which we created earlier.
4. For **Trigger**, click on the **trigger icon** ⚡ and browse for **Date/Time Triggers** > **Every Minute**.
5. In the **Data Storage Schedule** section, set **Recur every** to one hour.
6. Set the **Starting at** time to be on the hour (such as 1:00, 2:00, etc.), and make sure it is in the past. (The exact date does not matter.)

Note: The Data Storage Schedule determines when a new table will be created in the data storage. With these settings, we will get a new table created once an hour, on the hour.

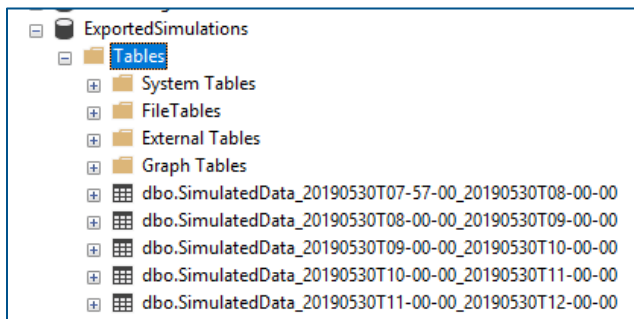


Figure 3 – Example table format

7. Go to the **Datasets** tab.
8. Select **Add** and browse for the **SimulatedData** dataset we created earlier.
9. Hit **Apply**.
10. Start the Hyper Historian Logger, if it was not already running.

Note: The data exporter supports online changes. A restart is not needed if data exporter changes are made when the logger was already running.

Once logging is started, within one minute you should be able to see new tables in your storage database.

Using Aliases

Datasets and storage connection strings can be aliased (or parameterized), allowing you to create one dataset or one storage and reuse them in multiple tasks to do slightly different things.

The following example will walk you through aliasing the folder of tags to export from the previous example.

1. Edit your **SimulatedData** dataset.
2. Go to the **Filters** tab.
3. Replace **Signals*** with **<<Path>>** (including the greater than and less than signs).
4. Apply your changes.
5. Edit your **SignalsTask** task.
6. On the **Datasets** tab you will see a specific field for **Aliases**. In that field, enter **Path="Signals*";**
7. Apply the changes.
8. Add a second task to the **Tasks** folder.
9. Give this task a unique name and configure it the same as **SignalsTask**, except in **Aliases**, enter **Path="ScanRates*";**
10. Apply the changes. Observe that you now have two tasks creating two sets of tables for two different folders of tags.

If you have multiple aliases in your dataset or storage you can specify them all in the task by separating them with semicolons, such as:

```
Alias1=Value1;Alias2=Value2
```

Note: Quotes around the alias value are optional. You may wish to use quotes if your value includes special characters, such as equal signs and semicolons.

The following fields can accept aliases:

- Dataset > Filters tab > Data Point Name
- Dataset > Filters tab > Point Name Filter
- Dataset > Filters tab > Path Filter
- Storage > Connection String

Storage Configuration

These sections will guide you to finding the correct connection information for your storage type and configuring that information in your Storage object in Workbench. Other than the connection configuration, the steps in the "Example: Export to SQL Server Database" section apply for these connection types as well.

Microsoft Azure SQL

For a Microsoft Azure database, go to your Azure and browse for SQL database. Under **Settings** you should see a **Connection string**.

Copy your **Connection String** and fill in the login credentials.

Create a new storage in Workbench, set the **Connection Type** to **SQL / Azure SQL**, then paste your connection string into the **Connection String** field. The final connection string should look something like this:

```
Server=tcp:AzureDatabaseName.database.windows.net,1433;Initial Catalog=CatalogName;Persist Security Info=False;User ID=username;Password=password;MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;
```

Microsoft Azure Data Lake

To export to an Azure Data Lake, create a new application registration (legacy) under your **Azure Active Directory**. Go to your Data Lake and **Add Access of this Application Registration to Data Lake**. Full permissions are required.

Note: See Microsoft Azure documentation on how to create a data lake. The exact steps on configuring a data lake may change.

Add **Access Control** for **Application** to your subscription. The Reader role should be enough.

In **Workbench**, create a new storage and set its **Connection Type** to **Data Lake**. Configure the **Storage Connection** section like so:

Workbench field	Where to find the value in Microsoft Azure
Storage Name	Data Lake Name under Data Explorer
Root Folder	Data Explorer (under Data Lake Name)
Domain Name	Azure Active Directory > Domain Names
Application ID	Azure Active Directory > App registrations
Authentication	Registered App > API Access > Keys

Note: These locations depend on Microsoft's organization of Azure and may change over time. Please consult Microsoft's documentation if you are having trouble finding these values.

Apache Kafka

For information about Kafka, see: <https://kafka.apache.org/intro>

This information is required to configure a storage for Kafka:

- Topic (created on Kafka side)
- Bootstrap Servers (server:port)

An example connection string might be:

```
BootstrapServers=localhost:9092;
```

```
Topic=DataExporter;IncludeDatasetInfo=true;
```

Here is an example Hyper Historian sample imported into Kafka:

```
{
  "Name": "SimpleDataset",
  "Start": "2019-07-25T15:27:01.273Z",
  "End": "2019-08-01T15:27:01.273Z",
  "Data": {
    "Point Name": "data:Signals.RampFast",
    "TimeStamp": "2019-07-26T15:48:14.589Z",
    "Status": 0,
    "Value": 77.858333333333334
  }
}
```

Apache Hadoop

For information on Hadoop setup please refer to online resources such as:

<http://hadoop.apache.org/docs/stable/index.html>

Connecting to Apache Hadoop requires the creation of a folder for the Hyper Historian data. In our example, we will create a folder named **HyperHistorian** using the following HTTP PUT request on Hadoop's [WebHDFS](#) (REST API):

http://server:50070/webhdfs/v1/HyperHistorian?user.name=use
rname&op=MKDIRS

Where *server* is the Hadoop hostname or IP address, and *user* is the Windows username.

Hadoop can operate on more than one port. Check which port Hadoop is using. The default ports are 50070 and 9870.

Once Hadoop is running and the HyperHistorian folder is configured, configure a storage object as follows:

Workbench field	Value
Connection Type	Hadoop File System
Root Folder	/HyperHistorian or your Hadoop folder name, if different
User Name	The username of your dedicated Windows user
Password	The password of your dedicated Windows user
Host	The location of your Hadoop server on the network
Port	50070 or 9870 (or other port that Hadoop has been configured to use)

The Connection String should populate automatically, and should look something like this example:

```
RootFolder="/HyperHistorian";Host="localhost";
Port="50070";UserName="Admin";Password=●●●●●●●●
●●●;
```

To see the data that has been exported, open a browser, navigate to your Hadoop hostname and port number (example: **localhost:50070**), go to **Utilities** and browse the file system. Under the HyperHistorian folder should be a new folder with

the name of the dataset. And under that should be CSV files with names such as: 20190725T15-27-01_20190801T15-27-01.csv.

The CSV contents can also be viewed via an HTTP GET call. Here is an example:

`http://server:50070/webhdfs/v1/HyperHistorian/dataset/csv-
file-name.csv?op=OPEN&user.name=username`

The reply should be the content of your CSV, which should look like this example:

Point Name,TimeStamp,Status,Value

```
data:Signals.RampFast,20190726T10:01:46.152,0,
40.88333333333333
```

```
data:Signals.RampFast,20190726T10:01:46.401,0,
42.96666666666667
```

```
data:Signals.RampFast,20190726T10:01:46.651,0,
45.05
```

```
data:Signals.RampFast,20190726T10:01:46.903,0,  
47.13333333333333
```

Local CSV files

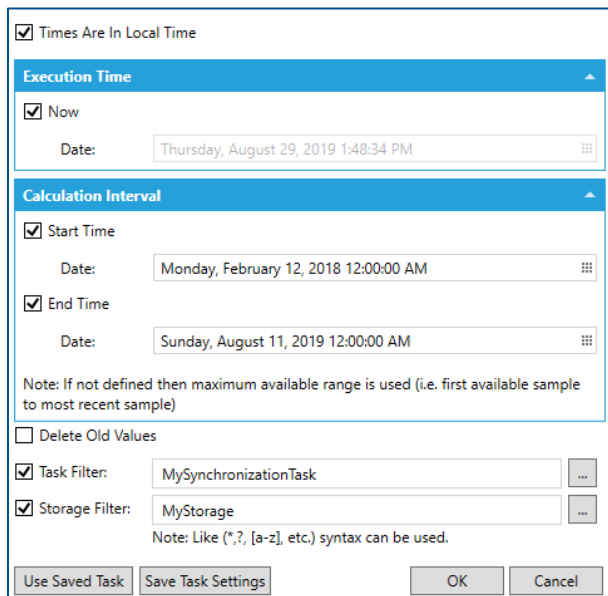
To export data locally to CSV files, configure a storage with the **Connection Type** of **Local CSV files**. In the **Root Folder** field, type or browse for your folder location. The **Connection String** field should be filled automatically.

Hyper Historian will automatically create new folders you're your dataset name in this location, which should contain your CSF files. An example CSV file name is: 20190725T15-27-01 20190801T15-27-01.csv

Manual (On-Demand) Synchronization Tasks

On-demand synchronization tasks allow the user to export any time period for a specific task or storage. This is beneficial for users who have already existing data which they would like to synchronize.

On-demand synchronization tasks can be configured from **Workbench** under **Historical Data > Hyper Historian > System Administration > Data Exporter Tasks Management**.



The dialog box for Data Exporter Task Management. It includes sections for Execution Time (with a 'Now' checkbox and a date field), Calculation Interval (with 'Start Time' and 'End Time' checkboxes and date fields), and Task/Storage filters (with checkboxes and dropdown menus). A note at the bottom states: 'Note: If not defined then maximum available range is used (i.e. first available sample to most recent sample)'. At the bottom are buttons for 'Use Saved Task', 'Save Task Settings', 'OK', and 'Cancel'.

Figure 4 – Data Exporter Task Management dialog

Exporting Multiple Values to Single Dataset

The Data Exporter is designed to work with single points at a time, where each row or record represents one point and one sample, but if needed, the user can utilize calculated tags to record several points in a single record. This can result in a more report-like set of data, with a single timestamp column and multiple value columns, one for each tag.

This example will walk you through creating and exporting an array of values using a calculated tag.

1. Right click on **Data Collections** or a specific folder underneath and select **Add Calculated Tag**.
2. Name your calculated tag, **CalculatedDataset-Index**.
3. In the Triggers section, select a trigger to determine how often the tag will be calculated. This will determine how often a record will be created for your set of samples.
4. Select the **Expression** tab.
5. Create array of values using the following expression:

```
array({{data:Signals.RampFast}},  
{{data:Signals.Ramp}}, {{data:Signals.RampSlow}})
```
6. On the **General** tab, set **Data Type** to **Float64** and check **Is Array**.

7. Apply your changes.
8. Under **Hyper Historian > Data Exporters > Datasets**, create new dataset called **DatasetArray**.
9. On the **Filters** tab, under **Tags**, select the **CalculatedDataset-Index** tag created earlier.
10. On the **Columns** tab, select the existing **Value** column and configure it as follows:
 - a. **Name:** RampFast
 - b. **Column Value Type:** Array of Values
 - c. **Index:** 0
 - d. **Data Type:** Float64
11. Add a second column and configure it the same as above, except use the name, **Ramp**, and the **Index** of **1**.
12. Add a third column and configure it the same as above, except use the name, **RampSlow**, and the **Index** of **2**.
13. Apply your changes.
14. Under **Hyper Historian > Data Exporters > Tasks**, create a new task. Select your storage and a trigger of at least one minute. On the Datasets tab, select the **DatasetArray** we created above. Make sure to apply your changes when finished.

Wait for your data to be exported and check the result. You should have records similar to the following:

Results	Messages	Point Name	TimeStamp	Status	RampFast	Ramp	RampSlow
1		calc:IndexCalculated-Tag	2019-07-29 14:01:32.0000000	0	98.56666666666667	99.71333333333333	78.526
2		calc:IndexCalculated-Tag	2019-07-29 14:01:33.0000000	0	6.9	1.38	80.21333333333333
3		calc:IndexCalculated-Tag	2019-07-29 14:01:34.0000000	0	15.233333333333333	3.046666666666667	80.21333333333333
4		calc:IndexCalculated-Tag	2019-07-29 14:01:35.0000000	0	23.566666666666667	4.713333333333333	80.21333333333333
5		calc:IndexCalculated-Tag	2019-07-29 14:01:36.0000000	0	31.9	6.38	80.21333333333333
6		calc:IndexCalculated-Tag	2019-07-29 14:01:37.0000000	0	40.23333333333333	8.046666666666667	80.21333333333333
7		calc:IndexCalculated-Tag	2019-07-29 14:01:38.0000000	0	48.56666666666667	9.713333333333333	81.901
8		calc:IndexCalculated-Tag	2019-07-29 14:01:39.0000000	0	56.9	11.38	81.901
9		calc:IndexCalculated-Tag	2019-07-29 14:01:40.0000000	0	65.23333333333333	13.046666666666667	81.901

Figure 5 – Multiple Tags on 1 row