Adroit Technologies

Adroit 10 / MAPS 4

How to effectively log your data

Adroit 10

MAPS
MITSUBISHI ADROIT PROCESS SUITE

COPYRIGHT

Advanced Worx 112 (Pty) Ltd
20 Waterford Office Park
189 Witkoppen Road (c/o Waterford Drive)
Fourways
South Africa

www.adroittech.co.za

Version	Date	Author	Comment	Approval
1.0	12.06.2023	J. Munitich	Original	

Contents

1.How to effectively Log your Data

Like any SCADA system one of the essential requirements of Adroit is to log values, so that they can be retrieved for long term analysis and reporting purposes.

At first it would appear that this process is relatively simple, but the logging of values has evolved over the years into a complex sub-system that requires a guide such as this one for you to use this effectively.

For instance, in the past it was potentially necessary to log the same tag using different logging rates and log sizes and even different destinations so multiple log sets could be created within a single datalog Agent. Now these multiple log sets are no longer required due to subsequent improvements made to the datalogging sub-system.

When recording or logging the values contained in Agents in the Agent Server, the first decision you need to make is the logging method that best suits your logging requirements. Adroit provides the following methods of storing data:

> **IMPORTANT:** If rapid logging is required, then use the Datalog Agent to log to the propriety Adroit database.

- Proprietary Adroit log files: This logging method uses the Datalog Agent and has been specifically designed for the needs of the SCADA industry to provide improved HIGH-SPEED logging performance, typically achieving sub-second logging rates.
- You should always use this logging method if you need REAL-TIME trending of your data within Adroit itself, such as in a chart on a graphic form.
  Note: Unlike in previous versions of Adroit the logging to databases by the Datalog Agent is disabled by default in Adroit 10 to force users to use the DBLog Agent, which has been specifically designed for this task
- A table in an OLE DB database: This logging method has been superseded by the DBLog Agent and remains purely as a backwards compatibility requirement.
- Batch processing: In this case, use the DBAccess Agent, which is ideally suited for batch processing as it allows you to define your table structure and then map Agents to the various fields, so that you can log both TEXT and NUMERIC values separately to create a single record per batch and then requires that you specify the unique trigger to log this batch.

However, the DbAccess Agent is not suited to logging a large number of tags to a database as it requires that you specify a trigger for each record that you intend to log and does not provide any tag importing functionality.

- Bulk logging of NUMERIC or DIGITAL tags: In this case, use the DBLog Agent, which has been specifically designed to efficiently log large numbers of primarily NUMERIC (e.g., Analog) or DIGITAL tags to databases, by providing bulk triggering mechanisms and tag importing functionality and even housekeeping functionality to ensure that only the most current data is stored in the database.

However, the DBLog Agent is not suited to batch processing as it uses a predefined database table so it is more difficult to group the values that belong to the same batch, and **it can ONLY log TEXT by converting all the logged values to Strings**.

> **IMPORTANT:** Database logging methods are **MUCH SLOWER** than when using the Datalog Agent due to the inherent inefficiencies of the OLE DB interface, especially when interfacing with remote databases. In most real-world systems, a combination of proprietary and database logging are used to facilitate both high speed logging and short-term trending as well as long term logging and thus reporting etc.

1.1. When logging to proprietary Adroit log files

As previously stated, this is a fast-logging method facilitated by the Datalog Agent that is designed to provide improved logging performance, by using datalog (.LDG) files to store the active logged data and log backup (.LDB) files that store the historical data.

You should always use this logging method if all you need to do is to display (report) your data within Adroit itself, such as in a chart on a graphic form.

When configuring the Datalog Agent, firstly select the required data-logging **Mode**, which depends upon how this process value is being used as follows:

- Continuous: the default setting, where Adroit logs data values as and when the value changes and then ONLY when this value meets the conditions imposed by the specified Deadbands (see below for more details). This is the default setting as this mode typically applies to most process variables, whose values you want to record for trending etc.
- Demand: Adroit only logs a data value when the LogNow or Trigger is activated and then ONLY when this value meets the conditions imposed by the specified Deadbands (see below for more details).

  This mode typically applies to batch type processes, to prevent the logging of every change it receives during the execution of a batch and only log the value at the end of a batch, in this case:
- The Trigger edit field, allows a String tag (Agent.slot) to be specified, which is monitored for a zero to non-zero transition, at which time a log occurs (provided it complies with the . When this occurs, a log occurs provided this value meets the conditions of the specified deadbands.
- The Log Now button is provided for manually triggering the logging.
- Both: this is a hybrid mode that uses both the Continuous and Demand data logging modes.

1.2. Configuring the effective logging rate

It is important to understand that a value is only logged when it changes and then ONLY when this value meets the conditions imposed by the specified **Deadbands**, which can be used:

- to eliminate the nuisance values or noise that can clutter up a log file without providing any meaningful change in value or
- to reduce the number of records that you log for a value which effectively 'compresses' your log files, so you are able to store more data on the same storage medium.

So, the effective logging rate is determined by configuring the **Time**... and/or **Value** deadbands, as follows:

- Using the **Value deadband:** Values are ONLY logged if their change in value EXCEEDS the specified deadband value. Hence when the value of a logged point does not exceed the previously logged value by more than the deadband, the value is not logged.
- Using the **Time deadband:** One of the following **Compression Method** algorithms is used to determine which value is logged, if the logged value changes MORE THAN ONCE during the specified deadband period:

  IMPORTANT: When using the Demand **Mode**, you need to configure the **Support compression algorithm on demand logging** global datalogging setting via the **Advanced...** button, to use these Datalog compression algorithms.

  Note: To use a logging Compression Method, you need to use a **Time...** logging deadband that is greater than 0.

- **Actual**: the logged value is the value at the end of the specified **Time** deadband (the default).

- **Average**: a time-weighted average is maintained for all the changes that occur and this is logged at the end of the specified.
- **Minimum**: the lowest or minimum value is logged at the end of the specified **Time** deadband.
- **Maximum**: the highest or maximum value is logged at the end of the specified **Time** deadband.

If you are using the **Time...** deadband to compress your log file to store more data then the **Average** method is recommended, since it calculates a time-weighted average of all the values that were recorded during the Time deadband period and this average is then logged at the end of this period.

In other words, a 1-hour average (as opposed to a 1-hour interval raw log) will provide a more accurate representation of what happened in the plant, than the same raw log. Since the raw log value will only represent the instantaneous value in the hour, not the average for the hour.

It is important to know that a value is only logged when it changes and when this change in value meets the conditions imposed by the specified **Deadbands** because this means that the specified **Length...** of the datalog (.LDG) file is only an estimation, which assumes that this value will change substantially at the effective rate of logging determined by the deadbands, however:

- if the value changes less than the effective rate of logging, then this log file will contain data for more than the specified duration but
- if the value changes more often than the effective rate of logging, these changes will not be recorded.

The Datalog Agent allows for a maximum of 53000000 records. This means that the specified datalog **Length...** (or period) and **Time...** deadband combination may not exceed 53000000 records. This corresponds to a maximum of 1010.9 MB of data at 20 bytes per record, which means 613 days at 1 second or 30 days at 50 ms.

> **Note:** The minimum **Time...** deadband allowed is 10 ms which only provides a maximum **Length...** (or period) of 6.13 days of logged data.

Furthermore, you can use **Out of limit logging** to bypass this process and immediately log any out of limit values (values that are not within their specified operating range), by ignoring the effective rate of logging, which is specified by the **Time...** deadband and using the faster log rate specified by the **Out of limit logging** -> **Time...** setting instead.

In other words, as soon as the logged value that is being logged, enters or leaves its user-specified limit range it is logged unconditionally, and continues to be logged at the new time deadband as specified by the **Out of limit logging** time setting (**Time...**). This allows the Datalog Agent to catch short spikes whose duration is less than the normal time deadband logging interval (**Deadbands** -> **Time...**)

1.2.1.Usage Example

A customer wants to monitor his incoming voltage of 220 V.

Previously, without using out of limit monitoring: he would create a Datalog Agent to log this voltage tag at a logging rate of 1 second for a period of 1 week. So, in this case, this Datalog Agent is configured (**Length** = 7 days and **Deadbands Time** = 1 sec) to log all changes to the voltage at 1 second and always keep one week's worth of this data.

Using out of limit monitoring: since he is actually not that interested in logging this voltage, if it remains within its normal operating voltage range of 215-225 volts, as long as the voltage is within this value range, he is happy to log it at a far slower rate - say 10 seconds.

But when the voltage breaches this range, then he wants this Datalog Agent to log this value at 1 second intervals.
So, in this case, this Datalog Agent is configured (Length = 7 days and Deadbands Time = 10 sec and out of limit **monitoring Time** = 1 sec and **Upper limit** = 225 and **Lower limit** = 215).

> **Note**: In this case he will be able to catch short voltage spikes whose duration whose duration are less 10 seconds, the normal logging interval.

> **Note:** To disable out of limit data logging, set both of the limit values are zero (default values), then NO limit checking is done and the Datalog Agent operates normally by ONLY using the normal time deadband logging interval (**Deadbands** -> **Time...**).

1.3. Specifying the required logging period

Typically, the specified **Length...** of the datalog (.LDG) file should not exceed what is required for operational trending purposes. In other words, should an operator only need to see 24 hours of trend data to properly monitor and control his plant then the datalog size should only be set for 1 day.

> **Note:** Any datalogs that cause their .LGD file to be greater than 2 GB will fail to start and user is notified accordingly and an associated EventLog entry is also created.

When the allocated size of the datalog (.LGD) file is filled, it wraps (starts again at the beginning). At which time one or more datalog backup (.LGB) files are created to store this older (historical) data. The datalogging sub-system automatically retrieves this historical data from these files, when this data is required by trends (charts) or other requests for historical data.

> **Note:** To disable datalog backups for a particular Datalog, set the **backupPeriod** slot of this Datalog Agent to **Disabled**.

> **Tip:** You can use these datalog backup (.LGB) files to extend the potential amount of logged data without creating larger log files, since larger log files have the side effect of causing the Agent Server to take longer to start up. In this case, while the active log (.LGD) file can remain small, the log backup (.LDB) files can store historical data for much longer time periods.

These datalog backup (.LGB) files also prevent the need to multiple proprietary datalog sets with increased periods and larger time deadbands and also replaced the legacy backup capability provided by the Systemdatalog Agent.

When the associated .LGD file wraps, each Datalog Agent creates their own backup (.LGB) files
In addition to creating .LGB files for Adroit to retrieve historical data, you can also create .CSV files to enable third party applications to trend or report this data too, by checking the **Enable CSV backups** checkbox at the bottom of the Datalog Agent.

In this case these .CSV files also contain a month of data and are stored in the same folder as the .LGB backup files and adhere to the SAME naming convention. Each .CSV is formatted to create reports by storing the logged data in the following 3 column file structure: Date (yyyy/mm/dd HH:mm:ss), Value and Quality.

IMPORTANT: If .CSV files are being created, Adroit ONLY only looks for and uses the data in the .LGB files. Therefore, if you have deleted an .LGB file but still have the .CSV file for the same period, Adroit is unable to retrieve the historical data for this time period.

Note: These .CSV files are optional and are ONLY created for the Datalog Agent/s that have this option selected.

Since multiple datalog backup (.LGB) files and optionally .CSV files are created by each Datalog Agent, the **Delete backup files older than (days)** housekeeping feature (at the bottom of each Datalog Agent) ensures that the created .LGB (and .CSV) files older than a specified number of days (default 90 days) are deleted from the backup folder.

Note: If you want to keep data older than this specified housekeeping number of days, then ensure that you have a MANUAL backup procedure in place to ensure that you move these older .LGB files to the required folder. However, in this case Adroit is unable to access the data contained in the moved files!

1.4. Managing Datalog Agents and their proprietary datalog (.LGD) files

- If you export and delete Datalog Agents that use proprietary log (.LGD) files, ensure that you restart the Agent Server before re-importing - so that they can remove the areas that they currently use, since multiple Datalog Agents can potentially use a single LGD file.
- When increasing the logging period (**Length...**) then ensure that the Agent database (.WGP) file is saved and the Agent Server is restarted at least once to configure the associated datalog (.LDG) file.

  Tip: This is NOT required when reducing the logging period.

1.5. When logging to a table in an OLE DB database

As mentioned previously this logging method is ideally suited for logging slow moving data and NOT for high-speed logging and can use the following two Agents, depending upon your intended reason for storing the data:

IMPORTANT: These logging methods are **MUCH SLOWER** than when using the Datalog Agent due to the inherent inefficiencies of the OLE DB interface, in interfacing with remote databases.

1.6. Using the DbAccess Agent for Batch processing

Batch processing is an ideal use for logging to a table in a database because it is inherently slow moving as it only requires logging data at the end of a batch job or a shift.

As stated above the **DbAccess Agent** is ideally suited for batch processing because this Agent allows you to define your table structure and then map the required tags to the various fields, so that you can log both TEXT and NUMERIC values separately to create a single record per batch and then this Agent requires that you specify the unique trigger to log this batch.

This Agent can also be used for the exchange of batch-specific data from databases. For instance, by retrieving the required set points from a database before the batch is started.

However, the DbAccess Agent is not suited to logging many tags to a database as it requires that you specify a trigger for each record that you intend to log and does not provide any tag importing functionality.

Note: All DbAccess Agents require a scan point license to run. This license is checked prior to performing a DbAccess transaction for the first time (on a per Agent basis) and is only returned when the Agent is deleted.

These Agents require careful consideration in terms of both the number of Agents to be used and the amount of work that they are to do, as they are resource "hungry".

Hence, both the Agent Server memory and overall memory usage should be monitored during configuration to avoid the excessive usage of these resources. The CPU usage should be monitored while during operation to ensure that the computer is able to handle the additional load.

1.7. Configuring the DbAccess Agent for batch processing

The DbAccess Agent
only establishes one connection per database; so many DbAccess Agents may share the same connection if they transact with the same database.

IMPORTANT: If you are connecting to a database across the network and your Agent Server is running as a service, then ensure that the user profile that the Agent Server service uses to log onto the computer HAS the necessary rights to browse the network and connect to the remote database.

Check the **Connect to database** checkbox after specifying the required database, to allow the table to be selected from the database and then click the **Update** button.

Specify the name of the table that you want to log your batch data to in the Table name edit box, either by typing in the name or if there is at least one connection to the database identified in the **connection string, then the browse button can be used to select the required table, from the** list of tables available in the selected database.

While the DbAccess Agent is able to perform a number of transactions on a database, in this case all you typically use the default **Transaction Type** of **Insert row**, which does NOT require you to specify the **Identify row** edit field.

Click the "add" button to add the required number of tag-to-field mappings, which allows you to define the structure of the specified table in the specified database.

Tip: If necessary, you and specify the required format of the tag in this field in the **Field Format** column, by using the available **Field Format Specifiers** depending on the data type of the tag that is being assigned to the field.

Typically, you will ONLY want to specify a trigger for this batch job, so click the **Edit...** button in the **Transaction Scheduling and Triggering** group and then browse in the required **Trigger** tag and the **Condition** that must be met by this value (namely: when zero, when non-zero and when changed) before this batch is logged.

Tip: The **Details...** button opens a dialog that gives the status of the transactions and shows any database connection errors.

1.8. Using the DBLog Agent for the bulk logging of values to databases

The **DBLog Agent** has been specifically designed to efficiently log large numbers of primarily NUMERIC (e.g., Analog) or DIGITAL tags to databases, by providing bulk triggering mechanisms and tag importing functionality and even housekeeping functionality to ensure that only the most current data is stored in the database.

> **Note:** While the DBLog Agent can be configured to log STRING tags, it can only do this by **converting all the logged values to Strings**, so in this case it is better to limit this DBLog Agent to ONLY containing STRING tags.

In other words, instead of logging each numeric (e.g., Analog) and/or Digital value separately, you can simply specify a list of tags to log and either export/import this list from a csv file or browse in this list individually in the required DBLog Agent.

1.9. Typical usage scenarios for the DBLog Agent

- to store data in a database for historical or primary SCADA functions, that do NOT require real-time trending.
- to log data for an external application that uses an OLE DB database, such as MS SQL, to report on or analyse your data, such as:
- proprietary reports like the Adroit Report Suite, whose General Reports report pack includes standard reports on the DBLog Agent data, which provides basic trending and summarises your data in a structured way.
- to provide the source data for the SCADA Intelligence tool to analyse your data in detail (in this case you need to check the String logging checkbox)
  Note: While DBLog Agents typically log Numerical and Digital values or bits, they can be configured to convert ALL values to Strings, by checking the **String logging** checkbox - but this is NOT recommended for general use, since **it requires MORE SPACE to store String values compared to Numerical values** and this can also cause formatting problems.
  So in this case it is better for a DBLog Agent to ONLY contain STRING tags.

However, the **DBLog** Agent is not suited to batch processing as it uses a predefined database table so it is more difficult to group the values that belong to the same batch, and it can ONLY log TEXT by converting all the logged values to Strings.

1.10. Practical DBLog Agent recommendations

- As stated above this Agent is NOT suited to logging data at a high resolution (logging rates below a minute) and this also places an unnecessary strain on the database and requires increased database management. So, **aim for a minimum logging interval of 5 minutes**. Factors affecting your achievable logging interval are:

- Each Agent added to a **DBLog** Agent generates its own SQL transaction, so the maximum number of Agents per **DBLog** Agent depends upon their required logging interval (rate).

In other words, the number of Agents that you need to log determines their possible or achievable logging rate, in that the greater the number of Agents you choose to log, the lower their achievable logging rate.

- Your hardware configuration also affects your achievable logging interval, as you are typically able to log data at a higher resolution to a SQL Server that is located on a separate computer

than your SCADA Server, so that these database transactions do not affect your SCADA Server.

- Due to the different logging requirements of numeric (e.g. Analog) and Digital tags, it is recommended to separate the numeric from the Digital tags, so that one DBLog Agent only contains numeric tags and another only contains Digital tags.
- For ease of management **do NOT add more than 1000 Agents to a single DBLog Agent**, instead create additional DBLog Agents to share the load of the Agents being logged.
- While DBLog Agents typically log Numerical and Digital values or bits, they can be configured to convert ALL values to Strings, by checking the **String logging** checkbox - but this is **NOT recommended for general use**, since it requires MORE SPACE to store String values compared to Numerical values and this can also cause formatting problems.

This should typically ONLY be set when you need to log string values in bulk, such as when providing the source data for the SCADA Intelligence tool to analyse your data in detail.

1.11. Recommendations for selecting the correct Logging Method

The DBLog Agent provides several bulk logging methods, so it is important that you select the correct one and configure its related settings correctly to ensure that the relevant data is being logged.

- For ANY data that is being retrieved from a telemetry system use the **Log on change** method, since it is possible to miss this data when using the **Log all on interval** method.
- For NUMERICAL data that is being regularly logged:
  - either use the **Log all on interval** method, if all the specified tags that have been added to the DBLog Agent are being logged at a similar rate so that the specified **Logging interval** ensures that all the changes in values of these tags are logged.
  - Or use the **Log on change** method, in which ONLY the tags that change are logged, typically when their value or quality status changes.

For this logging method specify the required VALUE deadband for each tag to ensure that only meaningful changes in value are logged via the **Deadband** column of either the CSV file or the **Tag list**.

In other words, specify the MINIMUM Value which the value being logged must EXCEED for it to be logged. Since, if the change in value is LESS than this value then this value is NOT logged.

> **Note:** To prevent this logging method from logging the tags when their quality status changes, add the following registry value and set it to 1: DisableOnQualityChangeLogging.
> The **Log all on change** logging method logs all the listed tags if ANY of their tag values change, if you decide to use this method then also specify the VALUE **Deadband** for each tag to ensure that only meaningful changes in value are logged.

- For DIGITAL data that is being regularly logged:

Use the **Log on change** method, in which ONLY the tags that change are logged, typically when their value or quality status changes.

> **IMPORTANT:** When using this logging method for DIGITAL values, ensure that the value **Deadband is 0**, otherwise NO values will be logged.
> **Note:** To prevent this logging method from logging the tags when their quality status changes, add the following registry value and set it to 1: DisableOnQualityChangeLogging.

However, if you are trending these DIGITAL tags in a chart, then you ALSO need to create another DBLog Agent for these same Digital Agents that uses the **Log all on interval** method, whose **Logging interval** is set to the charting period to ensure that the current state of all these DIGITAL tags will be displayed. More details:

When charting DIGITAL tags, the chart ONLY displays the value of each tag for the time specified by the specific charting period.

But if you ONLY use the **Log on change** logging method, then if a DIGITAL tag does not change in value during this charting period, then no data is returned, even though it has an existing valid state.

This is why you need the other DBLog Agent for the same Digital Agents that uses the **Log all on interval** method, whose logging interval is set to the charting period, which ensures that the last known state of each DIGITAL tag will be displayed by the chart.

The following triggering methods are seldom used as they generally are used for batch processing:

1.12.Other important DBLog Agent settings

The following top checkboxes of the DBLog Agent configuration dialog affect the general operation of this Agent, as follows:

- The **Enable** checkbox allows you to start/stop this Agent from logging the listed tags.

  IMPORTANT: If this **Enable** checkbox is not checked then this Agent will NOT log tags.

- The **Disable on standby** checkbox allows you to disable this Agent when its Agent Server is part of a cluster and is in standby node - to prevent the logging of DUPLICATE tags.
- The **String logging** checkbox, when enabled logs the tag values as Strings and not numeric values; this also allows you to log other string slots (this will log the first 79 characters).

  Note: This is unchecked by default as it is NOT recommended for general use, since it requires MORE SPACE to store String values compared to Numerical values and this can also cause formatting problems.

This should typically ONLY be set when you need to log string values in bulk, such as when providing the source data for the SCADA Intelligence tool to analyse your data in detail. In this case it is recommended that you ONLY add string values to this DBLog Agent.

The **DBLog** Agent allows for the list of tags to be added manually through the Agent dialog one by one or in bulk by exporting and re-importing the list in csv file format.

1.13.Database Configuration

Use the following fields to specify both the OLE DB database and table that you want to log your tags to:

> **Note:** While these fields specify SQL, you are able to connect to any OLE DB compatible database.

- **SQL connection string**: typically click the browse button to the right of this field and use the dialog to create the required OLE DB connection string to the database where these tag values are to be kept.

By default, this field specifies **global**, which uses the **Global OLE DB connection** string of the Agent Server, which can be specified in the **Adroit Config Editor**.

SQL table name: specify (type in) the name of the table in this database to log these tags to.

> **Note:** While multiple DBLog Agents can log to the same table, this can cause database management complications, if you need the tags within specific DBLog Agents to be stored for different durations. See the **Housekeeping** section below...

1.14.Housekeeping

In order to help you ensure that you ONLY store the most useful data in the database, by default the DBLog Agent implements **Housekeeping**, so that any records older than 90 days are deleted, by default.

Housekeeping is simply a cleanup action triggered on the table, specified by the **SQL table name** field of the database identified by the **SQL connection string** field.
Modify this setting by increasing or decreasing the maximum period that records should remain in the table, depending upon the storage capacity you have available for your database and how useful this data becomes to you the older it gets.

> **Note1:** A value of 0 disables this housekeeping functionality, provided NO OTHER DBLog Agents log to the specified table that have configured housekeeping!

> **Note2:** By design, one additional day's worth of data is kept for you.

As mentioned previously **Housekeeping** is a cleanup action triggered on the specified table that a DBLog Agent logs its tags to. Furthermore, multiple DBLog Agents can log to the same table. Therefore, when multiple DBLog Agents log to the same table that have DIFFERENT **Housekeeping** periods configured, the SHORTEST period will always be used!

By way of explanation, consider the following Housekeeping configuration examples:

Furthermore, when performing housekeeping you can also specify an OPTIONAL **Backup folder**, which is used to backup the old data before it is deleted from the table.

If you implement this **Backup folder**, we recommend that this is located on a backup or external drive that will not affect your drive on which your database is installed.

> **IMPORTANT1:** This folder needs to exist on the computer where the database is installed (this is NOT the Agent Server computer when using a remote database); otherwise, HOUSEKEEPING WILL NOT HAPPEN!

IMPORTANT2: ALSO ensure that the specified folder name does NOT contain SPACES or housekeeping will ALSO NOT HAPPEN!

IMPORTANT3: ALSO ensure that the database has the necessary permissions needed to add these files to this specified folder or housekeeping will ALSO NOT HAPPEN!

This **Backup folder** therefore specifies a path that is relative to the computer on which the database is installed (this is NOT the Agent Server computer when using a remote database).

Note: Leaving the **Backup folder** field empty disables this backup functionality, which is the default setting.

When this backup folder is being used each DBLog Agent creates a daily flat cabinet (.CAB) file of the data that is to be deleted (by the SHORTEST **Housekeeping** setting if multiple DBLog Agents log to the same table).
These files are self-extracting SQL scripts, which you need to manually execute against the required SQL database, to import your data.

PLEASE NOTE: It is up to you to manage the files stored in this folder yourself.

Once the DBLog Agent has been configured and enabled, it creates the table specified by the **SQL table name** field, which has the following **predefined structure**:

In addition to this table the DBLog Agent also creates the **Adr_DBLog_AgentDetails** table, which is updated when: the Agent is started; or when its tag list is changed; and unconditionally every morning at 01h00.

This table contains all the Agents referenced by all DBLog Agents to provide more details about these referenced Agents, which can be used to provide more context about them for reports.

For instance, this provides their Agent type and Agent description and is especially useful for Analog Agents in providing their unit of measurement, their alarm limits and value ranges.

The DBLog Agent also creates the **Adr_DBLog_Housekeeping_Config** table, which contains a list of all DBLog Agent tables requiring housekeeping, which is required to perform this housekeeping without affecting the logging operations of the DBLog Agents.