Store and Forward

The store-and-forward system provides a reliable way for Ignition to store data to the database. In Ignition, systems such as Tag Historian and SQL Bridge (Transaction Groups) use store-and-forward to ensure that data reaches its destination in the database, and is stored in an efficient manner. The store-and-forward system can be configured in a number of ways, offering both memory buffering for performance and local disk caching for safe storage.

Note: Store-and-forward engines are automatically created for each Database Connection.

Primary Features and Benefits

The store-and-forward system offers a number of benefits over other systems that log directly to the database, such as:

- **Data loss prevention**
  Data is removed from the system only when the write to the database has executed successfully.

- **Guaranteed ordering**
  Data is forwarded in the same order that it arrived, even if a database connection is not currently available.

- **Enhanced performance**
  By first buffering the data in memory, the store-and-forward system can optimize writes, and prevent the originating systems from blocking. This means that the system is less likely to lose data samples in the event of system slow downs.

Store and Forward Data Flow

Although the system offers settings that can affect the pipeline, by default the data flow occurs as follows:

1. Data is generated in some system.
2. Data is placed in a memory buffer.
3. If not removed from memory buffer in some time (the **Write Time**), or if a certain amount of data accumulates (**Write Size**), it is placed in the local cache.
4. The data sink, based on a database connection, pulls data in first from the local store, and then the memory buffer, based on the **Write Time** and **Write Size** settings under Forward Settings.
5. If the data fails to forward, either due to an error in the connection or in the data itself, it is returned to the buffer or cache.
6. If the data errors out too many times, it becomes quarantined.
7. Quarantined data can be managed through the Gateway, and can be deleted or un-quarantined, once the error is resolved.

Understanding the Forward Triggers

Data is forwarded from one stage to the next based on the **Write Time** and **Write Size** triggers. These settings work as an either/or manner, meaning that if either of them is surpassed, the data is forwarded. One important point to note is that the **Write Size** setting influences the transaction size of similar data to be forwarded, and therefore can have a big impact on performance. As a result, the **Write Time** should normally be used as the controlling factor, with the **Write Size** set to something that will provide reasonable transactions, like 100.

Store and Forward for Reliability

The store-and-forward system settings, while seemingly limited, offer a good deal of flexibility in tuning. Different types of situations and goals will likely require different configurations.

When the safety of the data is a concern, the goal is to get the data stored to disk as quickly as possible in order to minimize risk of loss due to a power outage or system failure. The local cache plays a crucial role in this, allowing the system to store data locally for any amount of time until the remote database can accept it. This protects against network failures and database failures, as well.

By setting the **write size** and **write time** of both the local cache and forwarder to low values, the data spends less time in the memory buffer. While the memory buffer can be set to 0 to bypass it completely, this is not usually recommended, as the buffer is used to create a loose coupling between the history system and other parts of Ignition that report history. This disconnect improves performance and protects against temporary system slowdowns. In fact, it is recommended that for reliable logging, this value be set to a high value, to allow the maximum possible amount of data to enter the system in the case of a storage slowdown.

In This Section ...