Historian Storage Locations Reference

Manage where historian data is stored.

Historian Storage Locations (Reference) define the repositories where the FrameworX Historian module archives time-series data from historian tags and tables.

Storage Locations provide:

- Database connections for time-series data
- External historian integration
- Custom storage via script classes
- Store and Forward buffering
- Multi-database support

Each Storage Location serves as a destination for historian tables, enabling flexible data archiving strategies.

Understanding Storage Locations
Storage Types
Default SQL Database Configuration
Adding Storage Locations
Store and Forward
TagProvider Configuration
Script Class Storage
Connection Strings
Performance Considerations
Multiple Storage Locations
Troubleshooting
Best Practices Checklist

Understanding Storage Locations

The Storage Location defines where Tag data will be archived establishing a clear linkage between tags and the storage infrastructure. When configuring the Storage Location, you can use:

- SQL databases
- Script Classes
- Connections with Third-Party Historians, Like Canary and others.

When you set the Storage Location, you specify the storage system for Tag data.

By default, the platform uses an SQLite database. You can change the default database or add additional databases if you need to use different ones.

Storage Types

SQL Databases (Default)

Built-in database support with TagHistorian object:

- SQLite (default, embedded)
- SQL Server
- PostgreSQL
- Oracle
- MySQL
- MS Access
- Firebird

TagProviders

External historian systems:

- **CanaryLabs** - Industrial historian with compression
- **InfluxDB** - Time-series database
- **GE Proficy** - Enterprise historian

Script Classes

Custom storage implementations:

- User-defined archiving logic
- Custom data transformations
- Third-party integrations
- Cloud storage adapters

Default SQL Database Configuration

Modifying Default Database

1. Navigate to **Historian Storage Locations**
2. Click **Settings icon** ??
3. Configure:
    - **Provider**: Database type
    - **Database**: Specific database
    - **Server IP**: Connection endpoint
4. Test connection with credentials
5. Click **OK**

Supported Providers and Databases

Provider	Supported Databases
SQLite	SQLite
Odbc	Firebird, MSAccess2003, MSExcel2003, MySQL, ODBC_DSN, ODBC_FileDsn, Oracle, SQLServer, SQLServer_WA
OleDb	CSVFile, MSAccess2003/2007, MSExcel2003/2007, Oracle, SQLServer, SQLServer_WA
OracleClient	Oracle
SqlClient	SQLServer, SQLServer_WA
PostgreSQL	PostgreSQL

Adding Storage Locations

Configuration Steps

1. Go to **Historian Storage Locations**
2. Click **Plus icon** ?
3. Configure properties (see table below)
4. Click **OK**

Storage Location Properties

Property	Description	Required
Name	Unique identifier	Yes
Description	Documentation text	No
StoreAndForward	Local buffering on connection failure	No
Storage System	TagProvider or Script Class	Yes
Object	Selected provider/class instance	Yes
Protocol	Communication protocol (TagProvider only)	Conditional
Station Editor	Connection string configuration	Conditional

Store and Forward

Operation

When enabled, provides data integrity during connection failures:

1. **Connection Lost** - Data buffered locally
2. **Local Storage** - Buffer database (SQLite by default; overridable to any supported database by adding a Datasets DB named "StoreAndForward")
3. **Reconnection** - Automatic retry
4. **Data Transfer** - Batch synchronization
5. **Cleanup** - Buffer cleared after success

Configuration

```
StoreAndForward: True
Buffer Location: Local SQLite by default; override with a Datasets DB named "StoreAndForward"
Retry Interval: 60 seconds (with Keep Local Copy, the forward cycle runs every 5 seconds)
Batch Size: 250 rows
```


The Store and Forward buffer defaults to a local SQLite spool. To use a different engine, define a Datasets database named "StoreAndForward"; its Provider and ConnectionString then back the buffer.

Best Practices

- Always enable for critical data
- Monitor buffer growth
- Plan disk space for buffering
- Test failover scenarios

TagProvider Configuration

CanaryLabs

Industrial historian with high compression:

```
Protocol: CanaryLabs
Connection: Server=hostname;Port=12345
Features: Compression, Aggregation, Web API
```

InfluxDB

Open-source time-series database:

```
Protocol: InfluxDB
Connection: http://server:8086;Database=mydb
Features: High throughput, SQL-like queries
```

GE Proficy

Enterprise historian platform:

```
Protocol: GEProficy
Connection: Server=hostname;Username=user
Features: Enterprise integration, Analytics
```

Script Class Storage

Implementation Requirements

Script class must implement:

```
public class CustomStorage
{
    public void StoreData(string table, DateTime timestamp,
                          Dictionary<string, object> values)
    {
        // Custom storage logic
    }

    public DataTable RetrieveData(string table,
                                  DateTime start, DateTime end)
    {
        // Custom retrieval logic
    }
}
```

Use Cases

- Cloud storage (AWS, Azure)
- Custom file formats
- NoSQL databases
- Message queues
- Data transformation

Connection Strings

SQL Server

```
Server=hostname\instance;Database=HistorianDB;
User Id=sa;Password=pwd;
```

PostgreSQL

```
Host=localhost;Database=historian;
Username=user;Password=pass;Port=5432;
```

Oracle

```
Data Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)
(HOST=server)(PORT=1521))(CONNECT_DATA=
(SERVICE_NAME=orcl)));User Id=hr;Password=pwd;
```

MySQL

```
Server=localhost;Database=historian;
Uid=root;Pwd=password;Port=3306;
```

Performance Considerations

Database Selection

Database	Best For	Limitations
SQLite	Small systems	10GB limit, single writer
SQL Server	Enterprise	License cost
PostgreSQL	Open source	Complex setup
InfluxDB	Time-series	Specialized queries
CanaryLabs	Compression	Proprietary

Optimization Tips

1. **Partition large databases** - By time or tag groups
2. **Index appropriately** - Timestamp always indexed
3. **Regular maintenance** - Archive old data
4. **Monitor performance** - Query execution times
5. **Size buffers correctly** - Store and Forward capacity

Multiple Storage Locations

Concurrent Storage

Configure multiple locations for:

- Redundancy
- Performance distribution
- Specialized storage
- Migration scenarios

Assignment Strategy

```
High-frequency data  Local SQLite
Production records  SQL Server
Compliance data  CanaryLabs
Temporary data  InfluxDB
```

Troubleshooting

Connection failures:

- Verify connection string
- Check network connectivity
- Confirm credentials
- Test firewall rules

Store and Forward issues:

- Check disk space
- Monitor buffer database
- Verify retry logic
- Review error logs

Performance problems:

- Analyze query patterns
- Check index usage
- Review table sizes
- Monitor network latency

Data not storing:

- Confirm table assignment
- Check storage location status
- Verify permissions
- Review trigger configuration

Best Practices Checklist

- [ ] **Test connections** - Before production deployment
- [ ] **Enable Store and Forward** - For critical data
- [ ] **Document configuration** - Connection strings and settings
- [ ] **Plan capacity** - Size databases appropriately
- [ ] **Monitor health** - Track connection status
- [ ] **Backup regularly** - Especially configuration
- [ ] **Use appropriate storage** - Match technology to requirements
- [ ] **Secure connections** - Encrypt sensitive data

In this section...
