Datasets Module Reference

Manage structured data sources and queries.

Datasets Module (Reference) provides comprehensive database connectivity, query execution, and data management capabilities for the FrameworX platform. The Datasets Module enables:

The Datasets Module enables:

- Multi-database connectivity
- SQL query execution
- Table-level CRUD operations
- File-based data exchange
- Store procedures support
- Real-time parameter embedding

The module serves as the data storage backbone for Retentive Tags, Alarms Logging, Historian, and custom application data.

Module Components
Configuration Workflow
Pre-Defined Databases
Runtime Architecture
Data Access Methods
Display Integration
Monitoring and Diagnostics
Security and Administration
Error Handling
Troubleshooting
Best Practices Checklist
Datasets Module Links

Module Components

Configuration Sections

Section	Path	Purpose
DBs	Datasets DBs	Database connections
Queries	Datasets Queries	SQL statements and procedures
Tables	Datasets Tables	Direct table operations
Files	Datasets Files	Recipe and text file management

Configuration Workflow

Step	Action	Location	Purpose
1	Define connections	Datasets DBs	Establish database links
2	Prepare queries	Datasets Queries	Create SQL statements
3	Map tables	Datasets Tables	Direct table access
4	Configure files	Datasets Files	Recipe management

Pre-Defined Databases

Four system databases included in every solution:

Database	Purpose	Location
Retentive	Retentive tag values	`<SolutionNameAndPath>.dbRetentive`
RuntimeUsers	Dynamic user management	`<SolutionNameAndPath>.dbRuntimeUsers`
AlarmHistorian	Alarm and audit records	`<SolutionNameAndPath>.dbAlarmHistorian`
TagHistorian	Time-series data	`<SolutionNameAndPath>.dbTagHistorian`

SQLite databases are created automatically if they don't exist. Other database types must exist before connection.

Protect the Runtime Users database. By default, the Runtime Users database is an unencrypted SQLite file in the solution folder, or in the Path set under Execution Environment in Runtime Startup. Anyone who can write to that folder can replace the file, and the users in the replacement file can log on. Move the Runtime Users database to a server database with its own logon, such as SQL Server with encryption turned on, in **Datasets DBs RuntimeUsers**. Restrict write access to the solution folder and that Path folder.

Runtime Architecture

Service Architecture

- **TRunModule.exe (Dataset)** - Module process
- **TServer** - Database access services
- **Client isolation** - No direct database access
- **Data virtualization** - Abstract data sources

Process Separation

TRunModule.exe (Dataset) handles:

- Reading all dataset configurations
- Does NOT directly connect to databases
- Consumes TServer database services
- Manages tag mapping and updates

TServer provides:

- Actual database connections
- SQL execution services
- Connection pooling
- Transaction management

Request Processing

1. Client request (display/script)
2. Server routing
3. Database execution
4. Result propagation
5. Tag mapping

Threading Model

Dataset Module Thread:

- Main coordination thread
- Manages request queue
- Handles tag mapping

TServer Database Threads:

- One per DB connection
- Sequential execution per thread
- Connection isolation

Client Request Threads:

- Asynchronous request handling
- Non-blocking UI operations
- Parallel client support

Data Access Methods

Queries

SQL statements with dynamic parameters:

sql

```sql
SELECT * FROM Production
WHERE BatchID = {{Tag Batch_ID}}
```

Tables

Direct CRUD operations without SQL:

- Automatic schema management
- WhereCondition filtering
- Tag mapping

Files

Text-based data exchange:

- Recipe management
- Configuration import/export
- XML/ASCII/Unicode formats

Display Integration

DataGrid Control

xml

```xml
<DataGrid DataSource="{Dataset.Query.ProductionQuery}" />
```

Script Access

csharp

```csharp
DataTable result = @Dataset.Query.MyQuery.SelectCommand();
@Tag.DataTableTag = result;
```

TK Methods

csharp

```csharp
TK.CopyDataTableToTag(result, @Tag.Template);
```

Monitoring and Diagnostics

Runtime Monitoring

Access **Datasets Monitor** during runtime:

- Operation status
- Row counts
- Error tracking
- Performance metrics

Namespace Properties

csharp

```csharp
int status = @Dataset.Query.MyQuery.LastStatus;
string error = @Dataset.Query.MyQuery.LastStatusMessage;
int rows = @Dataset.Table.MyTable.RowCount;
```

Security and Administration

SQL Injection Prevention

- Use parameterized queries
- Avoid string concatenation
- Validate inputs
- Use stored procedures

The platform's parameterization:

- Treats all parameters as values, not code
- Prevents malicious SQL execution
- Maintains data type integrity
- Supports all major database platforms

Network Gateway

Configure GatewayIP for secure routing access:

- Requires TWebServices
- Routes through gateway
- Isolates database

Time Zone Handling

- UTC storage default
- DateTimeMode configuration
- Local/UTC conversion

Error Handling

Error Detection

The module provides multiple error detection methods:

Property Monitoring:

csharp

```
@Dataset.Query.Query1.SelectCommand();
if (@Dataset.Query.Query1.Error != "") {
    // Handle error
}
```

Status Methods:

csharp

```
string status;
DataTable result = @Dataset.Query.Query1.SelectCommandWithStatusAsync(out status);
if (status != "OK") {
    // Handle error
}
```

Connection Failures

- Automatic retry logic
- Connection pooling recovery
- Error event propagation

Query Errors

- Exception capture in TServer
- Error property population
- Tag mapping skipped on error

Timeout Management

- Configurable command timeout
- Connection timeout settings
- Automatic cleanup

Troubleshooting

Common Issues

Connection Loss

- Check timeout settings
- Verify power settings
- Monitor network stability
- Review idle disconnect

Query Errors

- Validate SQL syntax
- Check table/column names
- Verify data types
- Test in database tool

Performance Issues

- Optimize queries
- Add indexes
- Reduce data volume
- Check network latency

Module Down

- Verify module running
- Check TServer status
- Review error logs

Database-Specific Syntax

Row Limiting Examples:

SQL Server:

sql

```sql
SELECT TOP 10 * FROM table
```

SQLite:

sql

```sql
SELECT * FROM table LIMIT 10
```

Oracle:

sql

```sql
SELECT * FROM table FETCH FIRST 10 ROWS ONLY
```

Oracle Limitations

NCHAR columns limited to 1999 bytes (auto-adjusted)

Best Practices Checklist

- [ ] **Error Handling** - Check LastStatus after operations
- [ ] **Connection Testing** - Use Test button before deployment
- [ ] **Query Optimization** - Design efficient SQL
- [ ] **Schema Respect** - Match DataTable to database
- [ ] **Case Sensitivity** - Use UPPER/LOWER for consistency
- [ ] **Backup Strategy** - Regular SQLite backups
- [ ] **Performance Design** - Consider all factors
- [ ] **Prefer asynchronous** - Use properties over methods
- [ ] **Avoid UI blocking** - No synchronous calls from screens
- [ ] **Use appropriate domains** - Client for user, Server for shared
- [ ] **Pool connections** - Reuse database connections
- [ ] **Handle errors** - Check status properties
- [ ] **Monitor performance** - Track execution times

Datasets Module Links

Explanation - to understand concepts

Modules / Business Operations / Datasets Module

Tutorials - to learn by doing

Tutorials / Business Operations / Datasets Module Tutorial

How-to Guides - to accomplish specific tasks

How-to Guides / Business Operations / Datasets Module How-to Guide

Reference - technical details

Technical Reference / Business Operations / Datasets Module Reference

---

In this section...
