# Datasets Monitor Reference

Monitor dataset operations and performance.



---

Datasets Monitor (Reference) provides real-time visibility into server-side database operations for Tables and Queries during runtime. The Datasets Monitor displays:

- Server operation status
- Row counts and results
- Error tracking
- Execution status
- Performance metrics

Monitor Table Properties
Runtime Property Access
Monitoring Scope
Using Monitor Data
Best Practices Checklist
Troubleshooting

Access via **Datasets** **Monitor** when connected to runtime.

---

## Monitor Table Properties

| Property | Description | Runtime Access |
|---|---|---|
| **Name** | Table or Query identifier | `@Dataset.Table.<Name>` |
| **SourceID** | Original configuration ID | `@Dataset.Table.<Name>.SourceID` |
| **RowCount** | Total rows returned | `@Dataset.Table.<Name>.RowCount` |
| **Disable** | Object disabled state | `@Dataset.Table.<Name>.Disable` |
| **LastStatus** | Operation result (0=success) | `@Dataset.Table.<Name>.LastStatus` |
| **LastStatusMessage** | Error description | `@Dataset.Table.<Name>.LastStatusMessage` |

---

## Runtime Property Access

Access monitor properties in displays or scripts:

### Table Properties

csharp

```csharp
// Get table status
int rows = @Dataset.Table.ProductionTable.RowCount;
int status = @Dataset.Table.ProductionTable.LastStatus;
string error = @Dataset.Table.ProductionTable.LastStatusMessage;

// Check for errors
if (@Dataset.Table.ProductionTable.LastStatus != 0)
{
    // Handle error condition
    LogError(@Dataset.Table.ProductionTable.LastStatusMessage);
}
```

### Query Properties

csharp

```csharp
// Monitor query execution
int queryRows = @Dataset.Query.MyQuery.RowCount;
bool isDisabled = @Dataset.Query.MyQuery.Disable;

// Validate results
if (@Dataset.Query.MyQuery.LastStatus == 0 && queryRows > 0)
{
    // Process successful query
}
```

---

## Monitoring Scope

### Server Operations Only

- Displays server-side executions
- Does not show client-initiated operations
- Tracks TServer database interactions

### Client-Side Monitoring

For client operations, use runtime attributes directly:

- Check status properties after execution
- Monitor from displays or client scripts
- Track individual client results

---

## Using Monitor Data

### Health Monitoring

csharp

```
// Check all tables health
bool allHealthy = true;
if (@Dataset.Table.Table1.LastStatus != 0 ||
    @Dataset.Table.Table2.LastStatus != 0)
{
    allHealthy = false;
    // Alert operator
}
```

### Performance Tracking

csharp

```
// Monitor row counts
if (@Dataset.Table.LargeTable.RowCount > 10000)
{
    // Consider pagination
}
```

### Error Logging

csharp

```
// Log all errors
public void LogDatasetErrors()
{
    if (@Dataset.Table.MyTable.LastStatus != 0)
    {
        string error = string.Format(
            "Table {0} Error: {1}",
            "MyTable",
            @Dataset.Table.MyTable.LastStatusMessage
        );
        WriteToLog(error);
    }
}
```

---

## Best Practices Checklist

- [ ] **Monitor regularly** - Check status periodically
- [ ] **Log errors** - Track all error messages
- [ ] **Set thresholds** - Alert on row count limits
- [ ] **Track performance** - Monitor execution times
- [ ] **Document issues** - Keep error history
- [ ] **Create dashboards** - Display key metrics

---

## Troubleshooting

**No data in monitor:**

- Verify runtime is active
- Check Designer connection
- Confirm server operations
- Review object configuration

**High error counts:**

- Check database connectivity
- Review SQL syntax
- Verify permissions
- Monitor database health

**Performance issues:**

- Check row counts
- Review query complexity
- Monitor network latency
- Optimize database indexe

---

### In this section...

---