

TCP Data Acquisition and Event Logging

By Anouar Moustarih, Technical Services Engineer, 4D Morocco

Technical Note 25-05

Table of Contents

| | |
|---|---|
| Table of Contents | 2 |
| Abstract..... | 3 |
| Introduction | 3 |
| Requirements | 3 |
| Class for Async Connection Lifecycle and Event Handling | 4 |
| Asynchronous Example | 4 |
| Explanation | 5 |
| Properties: | 5 |
| Methods: | 5 |
| Event Handlers (async):..... | 6 |
| Async Nature: | 6 |
| Handle Fragmented Data | 6 |
| Demo Application | 6 |
| Conclusion | 7 |

Abstract

This technical note explores the implementation of TCP data acquisition and event logging using 4D's newly introduced classes, TCPConnection and TCPEvent. The TCPConnection class provides a modern, asynchronous approach to managing TCP client connections, enabling seamless data exchange with external servers or devices. The TCPEvent class complements this by handling connection lifecycle events, such as connection establishment, data reception, and errors, allowing for robust event-driven programming. Through practical examples and detailed explanations, this note demonstrates how to leverage these classes to acquire data over TCP/IP networks and log critical events for monitoring and debugging. By integrating these features, developers can enhance the connectivity and functionality of their 4D applications, enabling real-time data acquisition and comprehensive event logging.

Introduction

In the realm of networked applications, the Transmission Control Protocol/Internet Protocol (TCP/IP) serves as the foundation for reliable data communication across the internet and private networks. TCP/IP ensures that data is transmitted accurately and in sequence, making it indispensable for applications that require consistent, error-free data exchange. For developers working on networked systems, understanding TCP/IP is essential, as it defines how data is packetized, addressed, transmitted, routed, and received.

With version 20 R8, 4D has enhanced its networking capabilities through the introduction of the TCPConnection and TCPEvent classes. These classes empower developers to establish and manage TCP connections directly within 4D applications, offering a straightforward way to integrate with external systems, services, or devices that communicate over TCP/IP.

The TCPConnection class allows developers to create TCP client connections to specified servers, supporting both synchronous and asynchronous modes of data exchange. It provides a range of callbacks that notify the application of significant connection events—such as connection establishment, data reception, and error conditions—enabling an event-driven approach to network communication.

Complementing this, the TCPEvent class encapsulates detailed information about each connection event. It includes data such as the event type (e.g., "connection," "data," "error"), the IP address and port of the remote system, and any received data. This information allows applications to respond intelligently to various network scenarios, ensuring robust and reliable TCP connection management—even when integrating with diverse external data sources.

Requirements

To fully understand and implement the concepts presented in this technical note on TCP Data Acquisition and Event Logging using 4D's new classes, TCPConnection and TCPEvent, readers should have the following prerequisites:

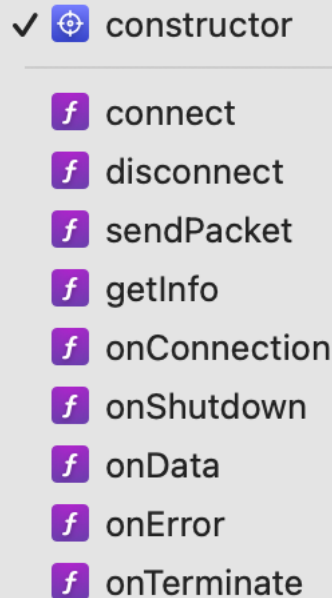
1. This note is based on 4D version 20 R8 or later, as the TCPConnection and TCPEvent classes were introduced in this release. Ensure that your 4D environment is up to date to utilize these features.

2. A basic understanding of the Transmission Control Protocol/Internet Protocol (TCP/IP) is essential. This includes familiarity with concepts such as IP addresses, ports, and the client-server model, as these are fundamental to establishing and managing TCP connections.
3. An internet connection is needed.

Class for Async Connection Lifecycle and Event Handling

Asynchronous Example

The following example defines a 4D class, which demonstrates how to manage a TCP connection to a server running on localhost, send commands, receive data, and handle connection events asynchronously.



```
property connection : 4D.TCPConnection
property url : Text
property port : Integer

Class constructor($url : Text; $port : Integer)
  This.connection:=Null
  This.url:=$url
  This.port:=$port

Function connect()
  This.connection:=4D.TCPConnection.new(This.url; This.port; This)
```

```

Function disconnect ()
    This.connection.shutdown()
    This.connection:=Null

Function sendPacket($blob : Blob)
    This.connection.send($blob)

Function getInfo()
    var $blob : Blob

    TEXT TO BLOB("Information"; $blob)
    This.connection.send($blob)

Function onConnection($connection : 4D.TCPConnection; $event : 4D.TCPEvent)
    ALERT ("Connection established")

Function onShutdown($connection : 4D.TCPConnection; $event : 4D.TCPEvent)
    ALERT ("Connection closed")

Function onData($connection : 4D.TCPConnection; $event : 4D.TCPEvent)
    ALERT (BLOB to text($event.data; UTF8 text without length))

Function onError($connection : 4D.TCPConnection; $event : 4D.TCPEvent)
    ALERT ("Connection error")

Function onTerminate($connection : 4D.TCPConnection; $event : 4D.TCPEvent)

```

Explanation

This 4D class elegantly handles asynchronous TCP connections. Here's a clear overview:

Properties:

- connection: Holds the TCP connection object (starts as Null).
- url: Server address (e.g., "127.0.0.1").
- port: Server port.

Methods:

- Constructor (\$url, \$port): Sets url and port, initializes connection as Null.
- connect (): Opens a TCP connection to the url and port.
- disconnect (): Closes the connection and clears connection.
- sendPacket(\$blob): Sends a binary data blob over the connection.

- `getInfo()`: Sends a blob with the text "Information".

Event Handlers (async):

- **onConnection**: Alerts "Connection established" when connected.
- **onShutdown**: Alerts "Connection closed" when disconnected.
- **onData**: Alerts received data as text.
- **onError**: Alerts "Connection error" on issues.
- **onTerminate**: Alerts "Connection terminated" when fully closed.

Async Nature:

Uses 4D. TCPConnection for non-blocking operations. Event handlers respond to network events (connection, data, errors) without halting the app.

Handle Fragmented Data

Account for fragmented data in *onData* callbacks, as TCP may split data across multiple events. When working with low-level TCP/IP connections, keep in mind there is no guarantee that all data will arrive in a single packet. Data arrives in order but may be fragmented across multiple packets.

Demo Application

The demo application accompanying this technical note demonstrates the functionality of the TCP connection management class and packet sending capabilities. It builds on the asynchronous example previously outlined to illustrate establishing a TCP connection, sending structured data, and managing asynchronous events within a 4D environment.

To execute the application, ensure an active internet connection and use 20R8 build 100353 or higher. The application employs 4D Internet Commands, such as `TCP_Listen`, to create a communication "socket" for connectivity.

In version 20 R9, a class called `TCPListener` supports the creation and configuration of a TCP server in 4D. Once instantiated, the `TCPListener` class enables receiving client TCP connections and facilitates communication through TCP-compatible protocols.

To run the application, adhere to the following steps in sequence:

- Ensure all prerequisites are fulfilled, then navigate to the application menu and select Run Application > Run.
- Once the sender windows appear, click on the "connect" button to establish a connection with the TCP server.
- After the connection is established, choose a sample file from the project resources folder ("Resources") using the "choose file" button.
- Upon successful file upload, hit the "send" button to transmit packets.
- Following successful packet reception, observe the progress bar displaying "End Receiving, ready for next packets..." and note the Receiver window displaying all transmitted packets.

- After complete packet reception, initiate transmission of new packets by uploading another sample file from the project resources folder.

Conclusion

The TCPConnection and TCPEvent classes in 4D v20 R8 have revolutionized TCP data acquisition and event logging, empowering developers to build robust, real-time networked applications with ease. Their asynchronous design and seamless integration with 4D's logging mechanisms deliver unmatched efficiency and reliability. As 4D continues to evolve, the promise of new TCP classes in future releases hints at even greater possibilities—pushing the boundaries of what 4D applications can achieve in the connected world. Stay tuned for these exciting advancements!