

OTEX FRAMEWORK

COMP7722 MAJOR PROJECT, SEMESTER 2 2016
MARK GILLARD (2065050 / GILL0235)

OTEX

AN OT-BASED TEXT EDITOR FRAMEWORK BY MARK GILLARD (2065050 / GILL0235)

CONTENTS

System Requirements	2
Launching OTEX Editor.....	2
Connecting to an OTEX server	3
Editing a document.....	4
Creating a dedicated OTEX server	5
Attributions.....	6
Source code.....	6
Appendix: OTEX Class Diagrams.....	7

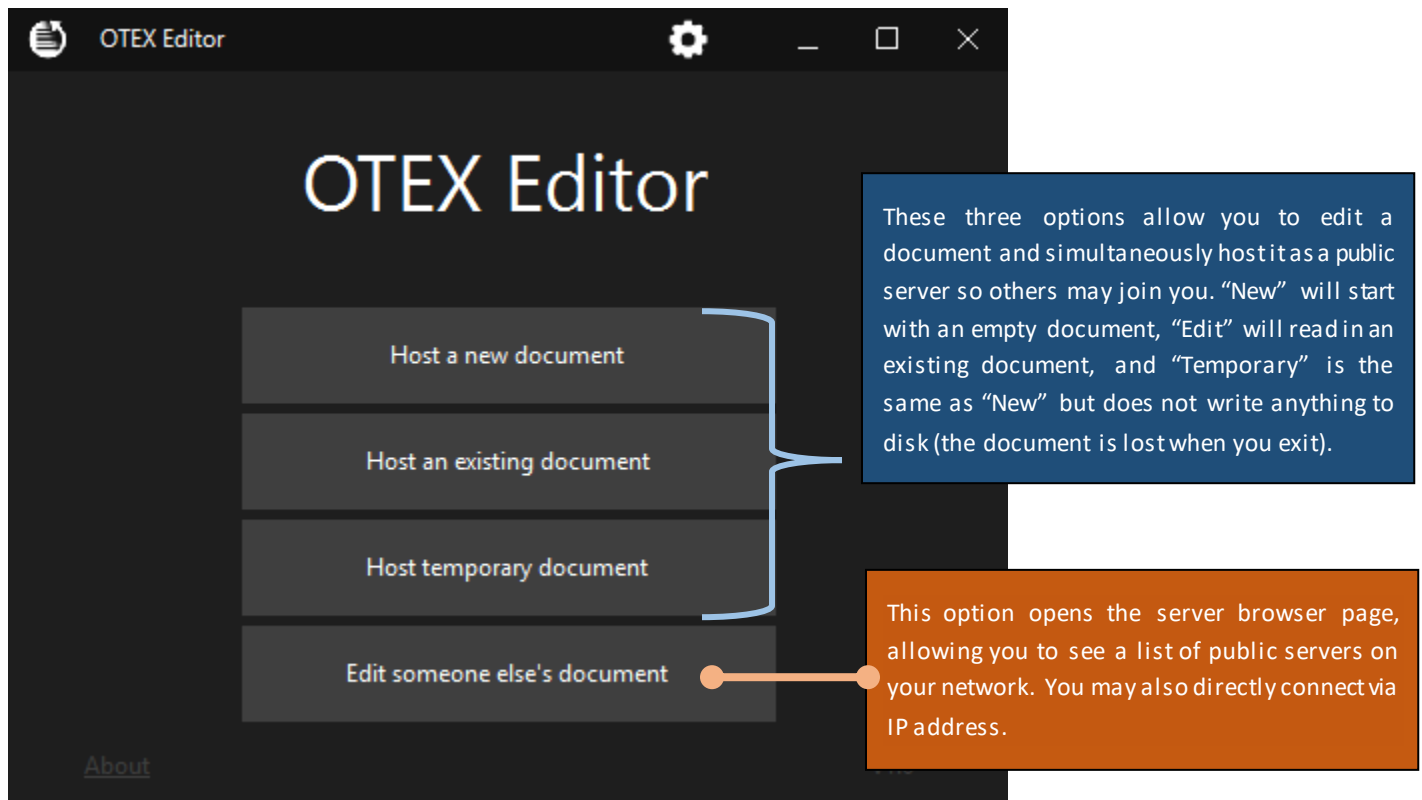
SYSTEM REQUIREMENTS

OTEX requires Windows 7 or higher, with the .NET Framework 4.5.2. Most systems that have not disabled windows updates will have this installed already, but if you need it you can download it here: <https://www.microsoft.com/en-au/download/details.aspx?id=42642>

LAUNCHING OTEX EDITOR

Run OTEX\x86\OTEXEditor.exe if your machine is 32-bit, or OTEX\x64\OTEXEditor.exe if it is 64-bit. If you're going to connect to an external server, you do not need to run the same platform's binary as the server; the communication between clients and servers is platform-agnostic.

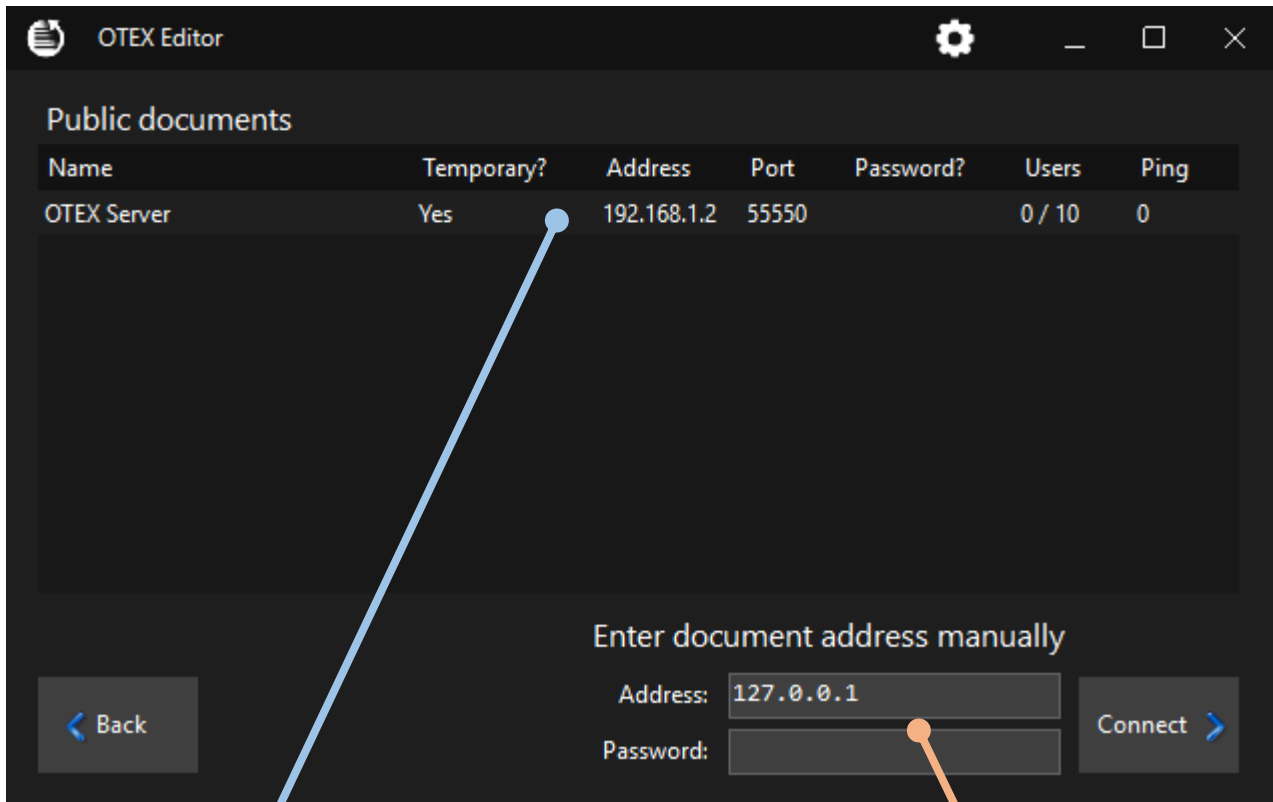
Once it's running, you'll see this:



- Selecting "New" or "Edit" mode will ask you to select a file to create/edit before the editor is launched.
 - **Note #1:** OTEX is a plain-text system, so attempting to force it to load binary files will have undefined results!
 - **Note #2:** OTEX Editor supports a wide variety of text file encodings on input, but writes files out in UTF-8 (without BOM).
 - **Note #3:** OTEX Editor supports three different line ending schemes: LF (unix), CR (?), and CRLF (windows). When files are first read OTEX Editor attempts to determine the scheme in use for the file by counting instances of each. If this fails it will fall back to the environment default (CRLF).
 - **Note #4:** Because of a limitation of the **FastColoredTextBox** control used by OTEX Editor, any tab (\t) characters in input files are replaced by 4 spaces.
- Selecting "Temporary" will launch the editor immediately.
- Selecting "Edit someone else's document" will open the server browser (see next page).

CONNECTING TO AN OTEX SERVER

The OTEX Editor server browser page allows you to connect to OTEX Servers:



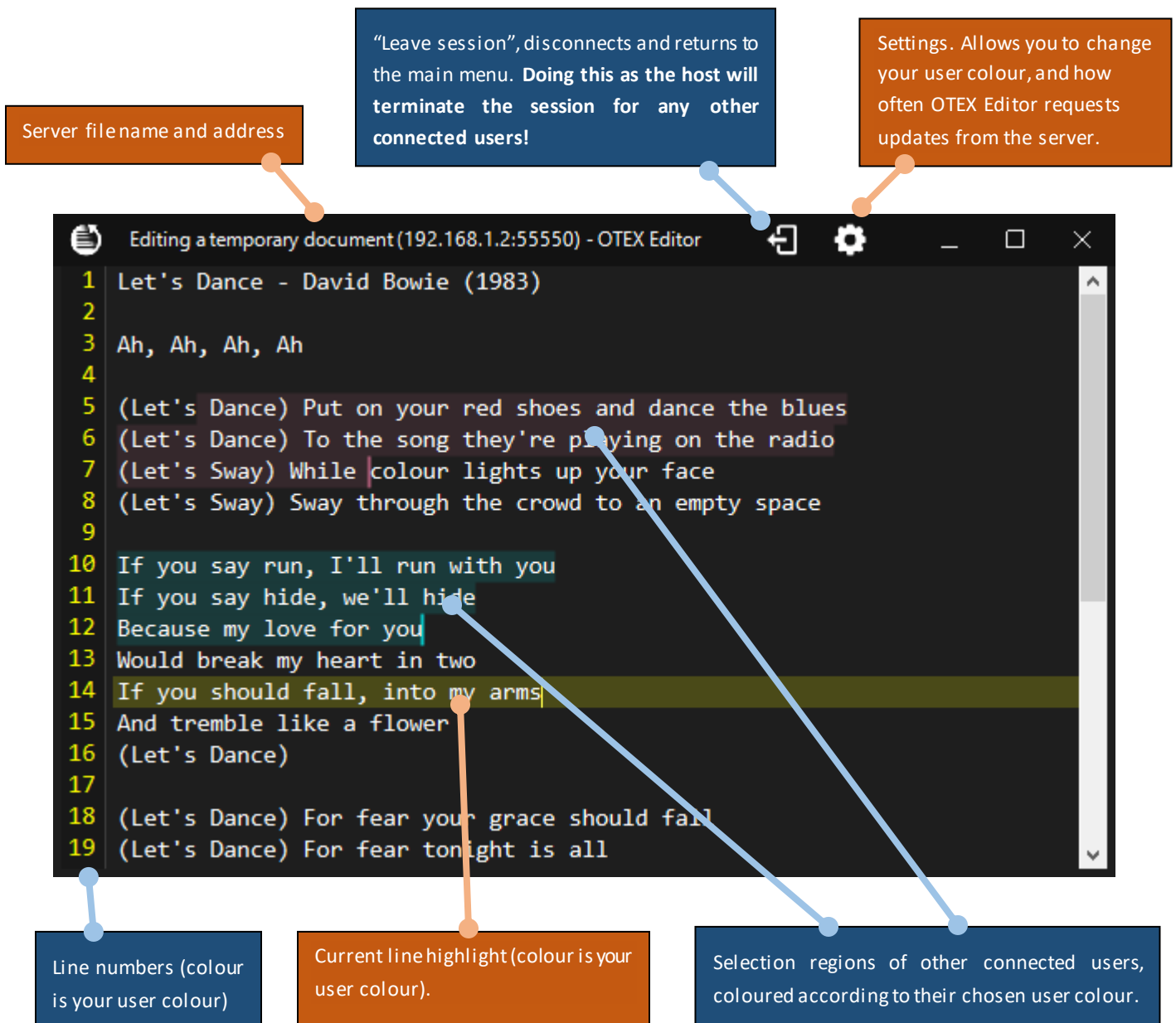
To connect to a public server, simply double-click it. If it requires a password you will be prompted for one.

To directly connect to a known server:

1. Type in the address in the "Address" field. If the server uses a non-standard port, indicate the port at the end of the address using :00000 notation.
2. Type the server's password into the Password field. If it does not require one, leave it blank.
3. Click Connect.

EDITING A DOCUMENT

Once connected to a server (or hosting one locally), you will see the editor interface:



Edit text as you would in any other text editor, and see changes made by other users reflected in the document as you go. The Editor supports a large number of IDE-like hotkeys:

Ctrl+F, Ctrl+R, F3	Find, Replace, Find Next
Ctrl+G	Go-To
Ctrl+Z, Ctrl+Y	Undo, Redo
Tab, Shift+Tab	Increase/decrease indent
Ctrl+U, Shift+Ctrl+U	Toggle uppercase/lowercase of selection
Ctrl+Shift+C	Toggles C-style comment prefix (//) of selected lines
Alt+Up, Alt+Down	Moves selected lines up/down
Shift+Del	Removes current line
Ctrl+B, Ctrl+Shift-B, Ctrl+N, Ctrl+Shift+N	Adds, removes and navigates bookmarks
Ctrl+Mousewheel, Ctrl+(NumpadPlus, NumpadMinus, 0)	Zoom
Ctrl+M, Ctrl+E	Start/stop macro recording, execute macro
Alt+F [char]	finds nearest[char]

CREATING A DEDICATED OTEX SERVER

It is also possible to host a server without launching the editor interface. To do so, use the command-line dedicated server.

Run `OTEX\x86\OTEXServer.exe` if your machine is 32-bit, or `OTEX\x64\OTEXServer.exe` if it is 64-bit. Connecting clients do not need to be using the same platform as the server version you are running; the communication between clients and servers is platform-agnostic.

By default, OTEX Server will host a private temporary file session on the default port (55550) for up to 10 users. Customize it with command-line parameters:

OTEXSERVER [file] [/EDIT|/NEW] [/PORT port] [/NAME name] [/PASSWORD pass] [/PUBLIC] [/MAXCLIENTS max] [/?]

file	Path to the plain text file to collaboratively edit. Omitting file path will create a temporary session (client edits will be lost when the server is shut down).
/EDIT	If a file already exists at the given path, edit it (do not overwrite with a new file). This is default.
/NEW	Opposite of /EDIT.
/PORT port	Port on which to listen for new OTEX TCP client connections. Can be any value between 1024-55560 or 55565-65535. Default is 55550.
/NAME name	A friendly name for the server. Limit of 32 characters (overflow is truncated).
/PASSWORD pass	Password required to connect to this server. Must be between 6 and 32 characters. Omit to allow clients to connect without a password.
/PUBLIC	Regularly broadcast the presence of this server to OTEX clients.
/MAXCLIENTS max	Maximum number of clients to allow. Defaults to 10, caps at 100.
/?	Prints help and exits.

Arguments may appear in any order. /SWITCHES and file paths are not case-sensitive.

ATTRIBUTIONS

OTEX Editor uses a very powerful RichTextBox alternative called **FastColoredTextBox**, which helps provides a lot of the more advanced features of the editor interface. FastColoredTextBox can found on [GitHub](#), [Nuget](#) and [CodeProject](#).

Icons used within OTEX Editor are variously sourced from websites like [Iconfinder](#) and [Flaticon](#), or created by hand. Any icons sourced from icon repository websites were selected based on “free for commercial use” licenses (i.e. I saved myself from having to remember to specifically credit every single one).

The OTEX Framework uses a form of Operational Transformation called Symmetric Linear Operational Transform, or SLOT, from the “NICE” approach. See:

Shen, Haifeng, and Chengzheng Sun. "Flexible notification for collaborative systems." *Proceedings of the 2002 ACM conference on Computer supported cooperative work*. ACM, 2002.

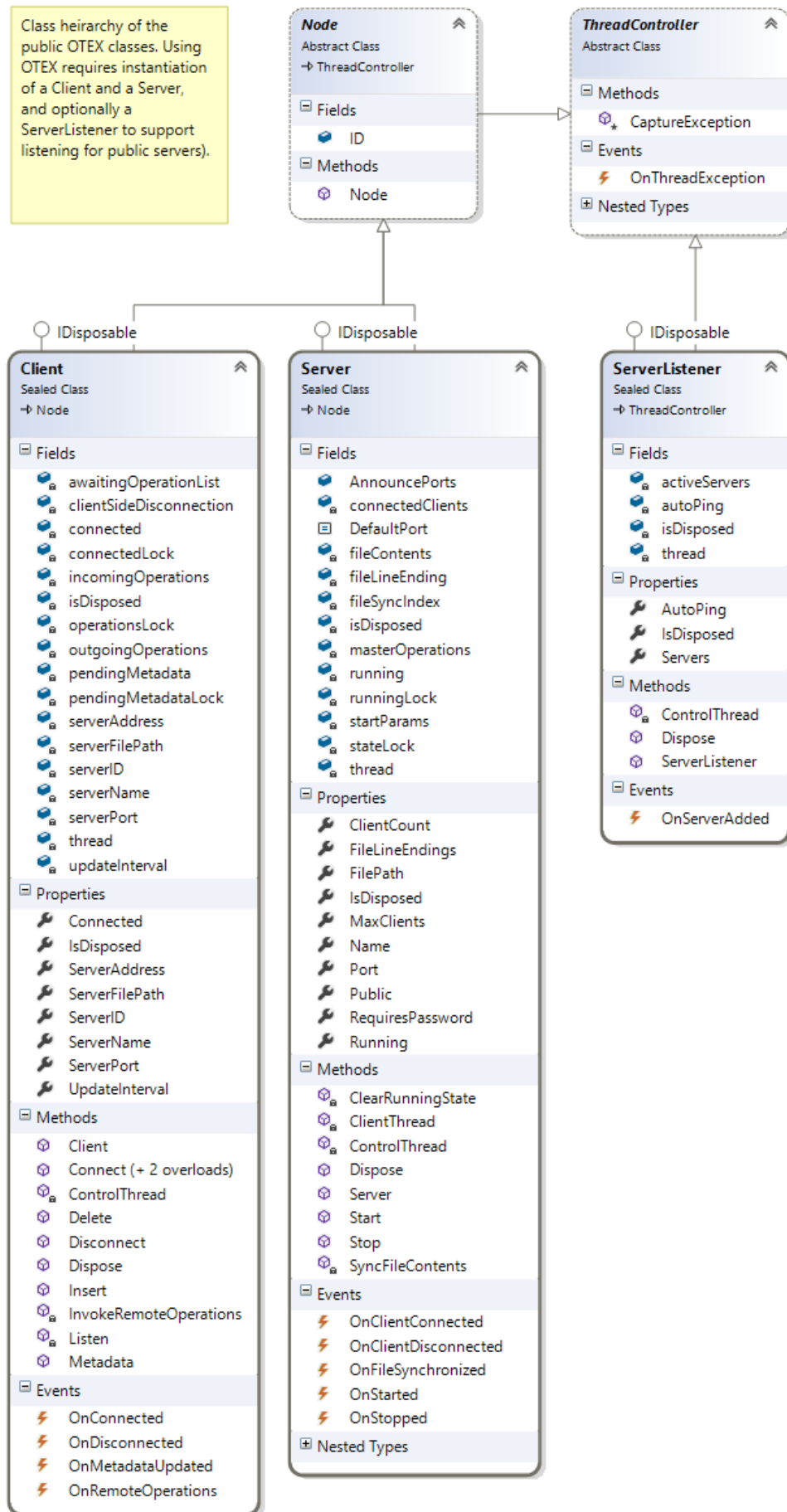
SOURCE CODE

The OTEX Framework is built in such a way that the Client, Server and ServerListener are all fully self-contained and implemented class objects, only requiring user code to “plug-in” to it by subscribing to the desired events. Case in point: the dedicated server application just an **extremely** thin command-line-wrangling wrapper around OTEX.Server. Similarly, the editor application is a relatively lightweight application too (though, any code requiring the manipulation of user interfaces is necessarily slightly more complex than a simple command-line application).

Code for the OTEX framework can be found on GitHub: <https://github.com/marzer/OTEX>.

Note for COMP7722: I have thoroughly commented the code, and specifically noted where something is relevant to the topic. These notes can be found by searching the C# files for the string “COMP7722”.

The code relies on a utility library (**Marzersoft.dll**), which is a personal library of useful classes, extensions and wrappers I’ve built over my years working with C#. It’s not open-source, but OTEX can be compiled by linking against the Marzersoft.dll files included with the OTEX distribution.



ServerDescription
 Sealed Class

Fields

- active
- clientCount
- endPoint
- id
- lastPingTimer
- lastUpdateTimer
- maxClients
- name
- ping
- pingThread
- port
- requiresPassword
- Tag
- temporaryDocument

Properties

- Active
- ClientCount
- EndPoint
- ID
- LastPinged
- LastUpdated
- MaxClients
- Name
- Ping
- Port
- RequiresPassword
- TemporaryDocument

Methods

- ServerDescription (+ 1 overload)
- Update
- UpdatePing

Events

- OnInactive
- OnUpdated

ServerDescriptions are sent out over UDP by Public servers, and captured by ServerListeners. At the server end these are treated as plain-old-data, but at the ServerListener end they are merged with new data and events are triggered notifying subscribers of updates to the server's state (e.g. the server administrator has changed the server's name).

PacketStream
 Sealed Class

Fields

- buffer
- client
- connected
- connectionCheckLock
- isDisposed
- lastConnectPollTimer
- stream

Properties

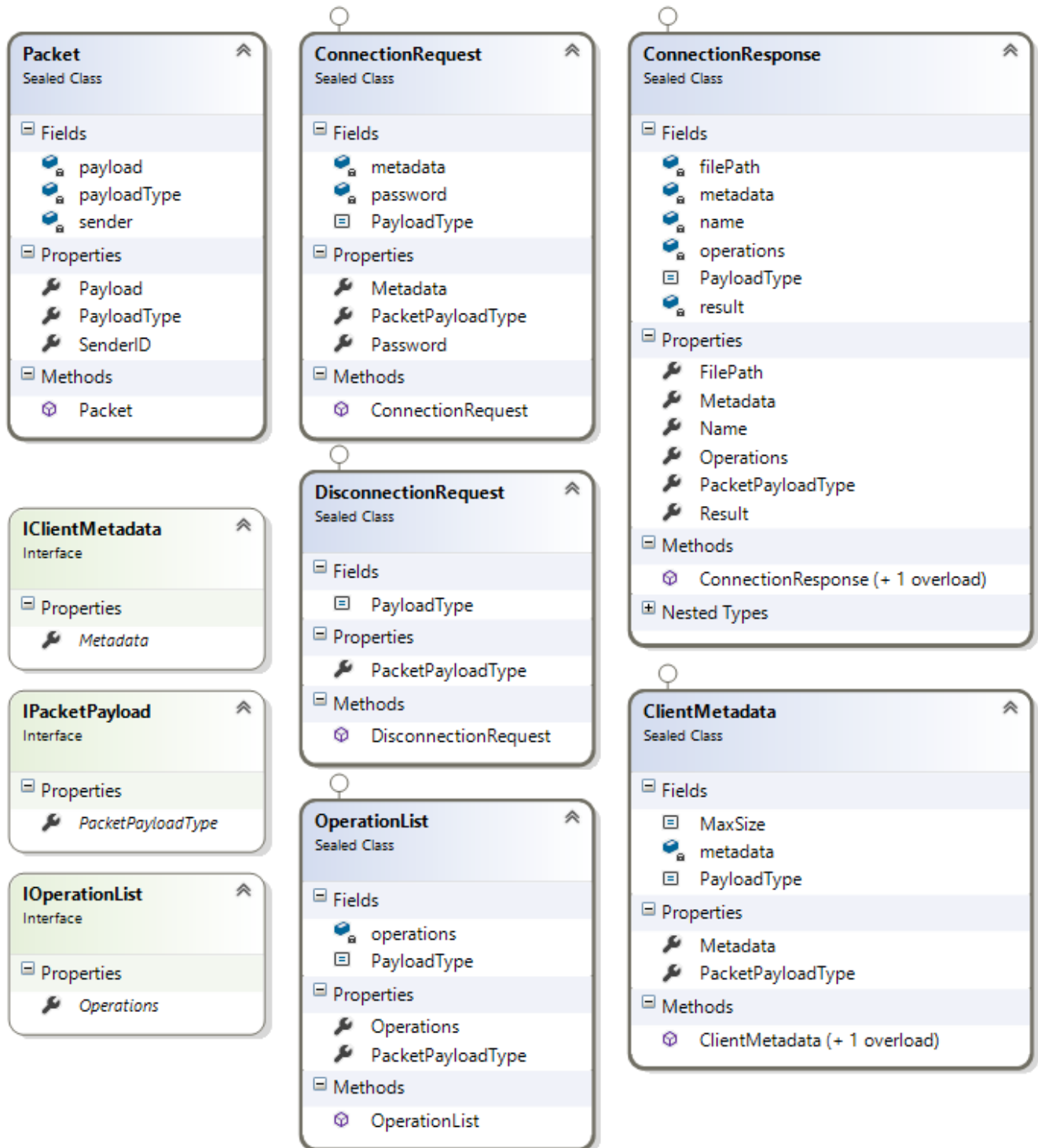
- Connected
- DataAvailable
- IsDisposed

Methods

- DetectBrokenConnection
- Dispose
- PacketStream
- Read
- Write<T>

PacketStream manages the state of a TcpClient connection and handles the serialization and deserialization of Serializable objects implementing the IPacketPayload interface.

this group contains of the different types of packets exchanged between the Client and Server over the normal TCP connection (via the PacketStream class).



OT Operation, with implementations of methods for NICE's SLOT algorithm.

Operation
Sealed Class

Fields

length

node

offset

text

Properties

IsDeletion

IsInsertion

IsNoop

Length

NodeID

Offset

Text

Methods

Execute

IT_DD

IT_DI

IT_ID

IT_II

MakeNoop

Operation (+ 2 overloads)

SymmetricLinearTransform

TransformAgainst

Password encryption class, so passwords are not transmitted in plaintext.

Password
Sealed Class

Fields

EncryptionKey

EncryptedPassword

Methods

Matches

Password

Port range helper class. Used for sanity checks and input validation.

PortRange
Sealed Class

Fields

first

last

Properties

Count

First

Last

Methods

Contains (+ 3 overloads)

PortRange

ToString