

The logo for Xb2.NET features the text 'Xb2.NET' in a bold, sans-serif font. The 'X' is white, 'b2' is red, and '.NET' is blue. The text is centered within a blue rectangular background that is slightly offset to the right, creating a layered effect.

**Web Server, SOAP Server,
& Secure Internet Library for Xbase++**

<http://www.xb2.net>
<news://news.xb2.net>

Powered by

Xb2.NET

Internet Library for Xbase++

- object-oriented sockets class
- generic Telnet server class
- FTP/FTPS client class
- HTTP/HTTPS server + client classes
- SOAP (Simple Object Access Protocol) server + client classes
- state/session management class
- XML parser
- SSL/TLS security via OpenSSL library

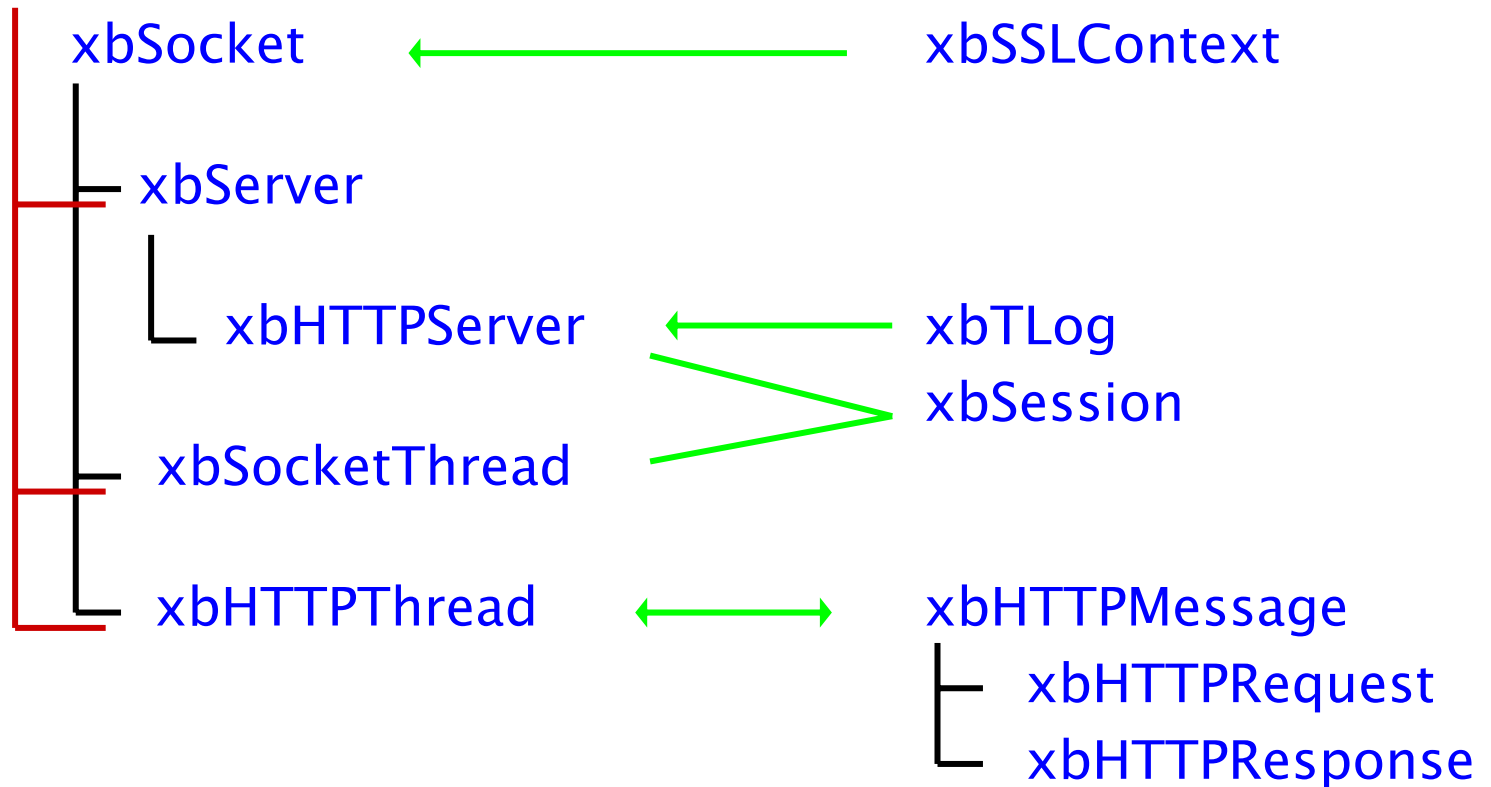
Powered by



Internet Library for Xbase++

Class Hierarchy (server)

Thread

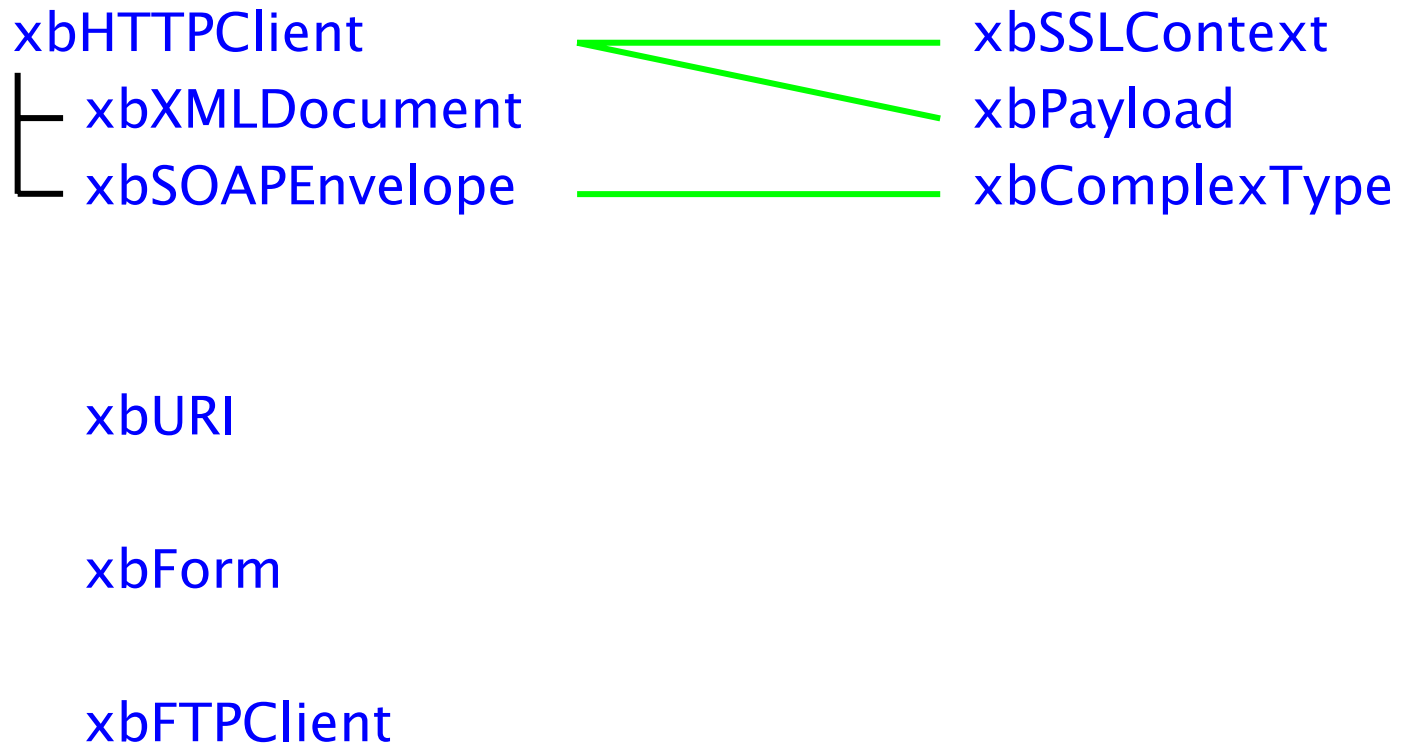


Powered by



Internet Library for Xbase++

Class Hierarchy (continued)



Powered by

Xb2.NET

Internet Library for Xbase++

xbSocket class

- foundation of Xb2.NET
- inherited by **xbServer**, **xbSocketThread**, **xbHTTPThread**
- secure communications via **xbSSLContext**
- each instance (object) => one socket connection
- each object knows about it's parent/children
- maintains statistics :RecvCount, :SendCount, :UpTime(), :StartTime,...
- codeblock events
 - :OnError = handle socket related errors
 - :Encode = spy/modify data prior to sending
 - :Decode = spy/modify data just after receiving
- must be explicitly destroyed when no longer needed!

Powered by

Xb2.NET

Internet Library for Xbase++

How to destroy a socket object?

```
FUNCTION SendToHost( cHost, nPort, cBuffer )
LOCAL oSock
oSock := xSocket():new(...)
IF ! oSock:Connect(cHost, nPort)
    ? "unable to connect to:", cHost, "on port:", nPort
    oSock := Nil
    RETURN .F.
ENDIF
IF oSock:Send(cBuffer) < 0
    ? "connection lost..."
    oSock := Nil
    RETURN .F.
ENDIF
// etc...
```

Xbase++ does not provide an automatic destructor method

Powered by

Xb2.NET

Internet Library for Xbase++

How to destroy a socket object (correct way):

```
FUNCTION SendToHost( cHost, nPort, cBuffer )
LOCAL oSock
oSock := xbSocket():new(...)
IF ! oSock:Connect(cHost, nPort)
    ? "unable to connect to:", cHost, "on port:", nPort
    oSock:Destroy()
    RETURN .F.
ENDIF
IF oSock:Send(cBuffer) < 0
    ? "connection lost..."
    oSock:Destroy()
    RETURN .F.
ENDIF
// etc...
```

Powered by

Xb2.NET

Internet Library for Xbase++

xbServer class

- inherited from `xbSocket`
- inherited by `xbHTTPServer`
- protocol-independent socket listener class
- automatically spawns a new `xbSocketThread` (*the worker thread*) when a client connection is accepted
- automatically keeps track of each connected client
 - `:ClientList` = array of active `xbSocketThread` objects
 - `:ActiveConnections()` = number connected sockets
- provides codeblock event slots:
 - `:OnConnect` = function to service client
 - `:OnMaxConnect` = what happens when there are too many connections
- can define custom client handler class by assigning new class reference to `:WorkerClass`

Powered by

Xb2.NET

Internet Library for Xbase++

xbSocketThread class

- inherited from **xbSocket** and **Thread**
- automatically spawned by **xbServer** when a connection with a peer is accepted
- inherits the parent's properties including **xbSSLContext**
- It's the "worker" thread that will service the client
- runs in it's own thread
- always have access to this object by calling Xbase++ ThreadObject() function.
- socket automatically closed and destroyed when thread terminates

Powered by

Xb2.NET

Internet Library for Xbase++

xbHTTPServer class

- inherited from **xbServer**
- HTTP/HTTPS 1.1 listener + secure SOAP server
- easy to install & use. Don't need WAA, Gateway, IIS, Apache, etc... Your Xbase++ app is the server
- because there are less layers, it's fast!
- can have multiple HTTP servers running simultaneously within same application. Each runs in it's own thread.
- automatically spawns a new **xbHTTPThread** (*the worker thread*) when a client connection is accepted
- automatically keeps track of each connected client
 - :ClientList = array of active **xbHTTPThread** objects
 - :ActiveConnections() = number of connected sockets

Powered by

Xb2.NET

Internet Library for Xbase++

xbHTTPServer class (continued)

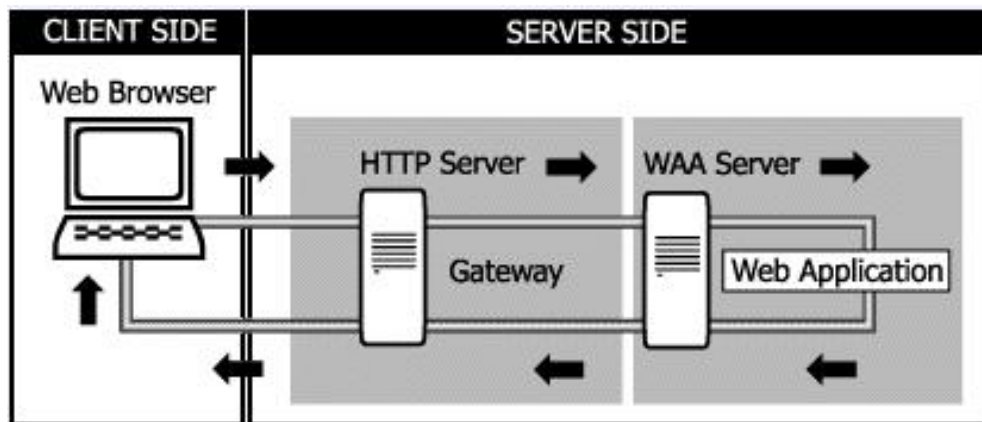
- automatic state/session management – simply save & restore data using :SetCargo() and :GetCargo()
- codeblock event slots (in addition to inherited ones):
 - :OnGET
 - :OnPOST
 - :OnPUT
 - :OnSOAP
 - :onDelete
 - :OnHTTPError
 - :OnInvalidCommand
 - :OnNotFound
 - :FilterRequest = spy/modify request before processing

Powered by

Xb2.NET

Internet Library for Xbase++

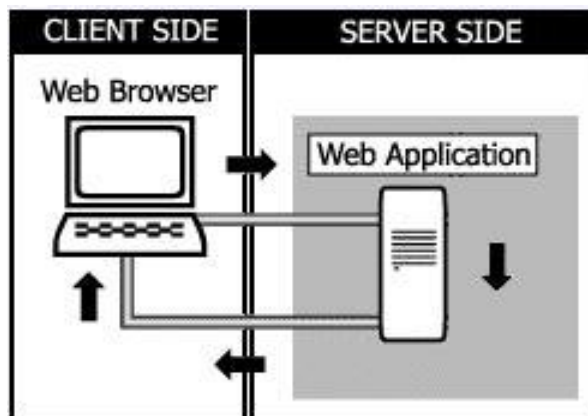
WAA Data Flow:



Required:

- Web Browser
- HTTP Server
- Gateway
- WAA Server
- Your Web App

Xb2.NET Data Flow:



Required:

- Web Browser
- Your Web Application

Easy to use

Easy to install

Easy to support

Powered by

Xb2.NET

Internet Library for Xbase++

xbHTTPThread class

- inherited from **xbSocket** and **Thread**
- automatically spawned by **xbHTTPServer** when a connection with a client is accepted
- It's the "worker" thread that parses HTTP request and generates appropriate HTTP response
- runs in it's own thread
- always have access to this object by calling Xbase++ ThreadObject() function.
- socket automatically closed and destroyed when thread terminates
- provides automatic error handling
- :GetVar() to retrieve parameters sent by client
- :SetCookie()/:GetCookie() to save/retrieve data on client
- :SetCargo()/:GetCargo() to save/retrieve data on server

Powered by

Xb2.NET

Internet Library for Xbase++

xbHTTPRequest class

- inherited from [xbHTTPMessage](#) and [xbURI](#)
- **client side**: used to compose and send request to an HTTP server
- **server side**: used to receive and parse a request transmitted by an HTTP client
- provides access/assign methods for all RFC-2616 (the HTTP/1.1 spec) header fields

Powered by

Xb2.NET

Internet Library for Xbase++

xbHTTPResponse class

- inherited from [xbHTTPMessage](#)
- **server side**: used to compose and send a response to a connected HTTP client
- **client side**: used to receive and parse a response transmitted by an HTTP server
- provides access/assign methods for all RFC-2616 (the HTTP/1.1 spec) header fields

Powered by

Xb2.NET

Internet Library for Xbase++

xbSession class

- HTTP is a **connectionless** & **stateless** protocol:
Clients connect to the server, make a request, get a response, and disconnect. **No information is maintained about a transaction after it has been processed**
- xbSession class provides ability of saving persistent data on the server across multiple HTTP requests
- sessions are automatically managed by the xbHTTPServer class; just use oThread:SetCargo() / oThread:GetCargo()
- xbHTTPServer periodically performs a :SweepSessions() which releases system resources allocated to sessions that have been idle for more than the specified :SessionTimeout

Powered by

Xb2.NET

Internet Library for Xbase++

Maintain session without cookies:

1. Use one of these methods to open session in each request:
`ThreadObject():OpenSession()`
`ThreadObject():SetCargo()`
`ThreadObject():GetCargo()`
2. Get the session ID by using
`ThreadObject():GetSessionHandle()`
3. Add a variable with the name "_SID" and a value returned by `:GetSessionHandle()` to each URL hyperlink that points back to the server.
4. Add a hidden variable with the name "_SID" and a value returned by `:GetSessionHandle()` to each form that will be posted back to the server.

Powered by

Xb2.NET

Internet Library for Xbase++

xbSession example w/o cookies:

```
FUNCTION WEB_Test1()
  oClient := ThreadObject()
  oClient:SetCargo("message", "Come here Watson")
  cID := oClient:GetSessionHandle()
  oClient:HTTPResponse:Content := "<html><body>" +
  "<a href=Test2?_SID="+cID+">Click here</a>" +
  "</body></html>"
  RETURN

FUNCTION WEB_Test2()
  oClient := ThreadObject()
  oClient:HTTPResponse:Content := "<html><body>" +
  oClient:GetCargo("message") +
  "</body></html>"
  RETURN
```

Powered by

Xb2.NET

Internet Library for Xbase++

xbURI class

- inherited by `xbHTTPRequest` class
- a Uniform Resource Identifier (URI) is a compact string of characters for identifying an abstract or physical resource
- a URI consists of the following components (some of which may not be present in all URI's):

`[scheme]://[authority][path]?[query]#[fragment]`

example: `http://www.xb2.net/xb2net.zip`
`ftp://ds.internic.net/rfc/`

- `xbURI` class allows creating, parsing, encoding and decoding of RFC-2396 compliant URI and URL strings

Powered by

Xb2.NET

Internet Library for Xbase++

xbURI class example

```
oURI := xbURI():new()
oURI:Scheme := "http"
oURI:Authority := "www.xb2.net"
oURI:Path := "/beta/download.htm"
oURI:Fragment := "Xbase 1.8"
oURI:SetVar("UID","Boris")
oURI:SetVar("SID","123%+&abc") // note special chars in value
oURI:SetVar("V+1","xbURI test") // note special chars in name
cHLink := oURI:AsString()
```

<cHLink> will contain the following URL encoded string:

```
http://www.xb2.net/beta/download.htm?UID=Boris&
SID=123%25%2B%26abc&V%2B1=xbURI%20test#Xbase%201.8
```

Powered by

Xb2.NET

Internet Library for Xbase++

xbFORM class

- a form is typically used to transport encoded data in the form of “name=value” pairs over the web
- class provides methods for creating, parsing, encoding and decoding of web forms
- Set and retrieve data using :SetVar() and :GetVar()

Powered by

Xb2.NET

Internet Library for Xbase++

xbSOAPEnvelope class

- SOAP is an open standard developed by the W3C that can be used in a distributed environment to execute platform and application independent remote procedure calls (RPC).
- class provides methods for creating, parsing, encoding, decoding and executing of XML SOAP envelopes, eg:

```
// create SOAP object to hold request
oRequest := xbSOAPEnvelope():new()
oRequest:Namespace := "urn:xmethods-CurrencyExchange"
oRequest:SetVar("Country1", "USA" )
oRequest:SetVar("Country2", "EURO")

// execute request & print result
// result is returned as another SOAP object
oResult := oRequest:Execute(
    "http://services.xmethods.net:80/soap", ;
    "getRate")
? "Exchange rate is:", oResult:GetVar("Result")
```

Powered by

Xb2.NET

Internet Library for Xbase++

xbSSLContext class

- context for implementing secure network communication
- select protocol to use (SSL v2, SSL v3, TLS v1)
- set allowable cipher suites
- load CAs (Certification Authorities)
- load certificates and private keys

```
// define SSL context
```

```
oSSL := xbSSLContext():new( SSL_v23_server )
```

```
oSSL:UseCertificateChainFile("MyCert.PEM")
```

```
oSSL:UsePrivateKeyFile("MyCert.PEM")
```

```
// attach SSL context to HTTP server instance
```

```
oHTTPS := xbHTTPServer():new( ;
```

```
    INADDR_ANY, 443, oSSL)
```

```
oHTTPS:start()
```

How to create certificate signed by public CA:

1. create CSR (certificate signing request) + private key :
`openssl req -new -nodes -keyout MyKEY.pem
-out MyCSR.pem -config xb2net.cnf`
2. backup **mykey.pem** and keep secret (don't give to anyone)
3. submit **mycsr.pem** to Certifying Authority (CA) (eg. VeriSign)
4. receive certificate from CA by email (certificate is digitally signed by your CA and contains your public key, your name, and the name of the CA)
5. create new text file, eg. **MyCert.PEM** as follows:
 - a) copy & paste text from mykey.pem
 - b) append certificate text received from CA (item 4 above)

To query certificate:

```
openssl x509 -in mycert.pem -noout -text
```