

Getting started with Habari XB Client

Michael Justin

A short guide for the first steps with the xmlBlaster client library

Trademarks

Java, JavaBean, JDK, Sun, Sun Microsystems, and the Sun Logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All Borland brands and product names are trademarks or registered trademarks of Borland. All CodeGear brands and product names are trademarks or registered trademarks of CodeGear. Microsoft, Windows, Windows NT, and/or other Microsoft products referenced herein are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other brands and their products are trademarks of their respective holders.

Contents

Introduction.....	3
About Habari XB Client.....	3
About xmlBlaster.....	4
License.....	5
Installation.....	7
Requirements.....	7
Upgrades.....	7
Architecture.....	8
THabariXBExpressBase Component.....	8
THabariXBExpress Component.....	9
Examples.....	10
Sending Messages.....	10
Subscribe Messages.....	12
HabariXBExpress.....	14
Events.....	14
Methods.....	14
Properties.....	15
Known Limitations.....	19
Message Exchange.....	19
Object Exchange.....	19
References.....	20
Release Notes.....	21
Version 1.2.....	21
Version 1.1.....	21
Version 1.0.....	21
Index.....	22

Introduction

About Habari XB Client

How Can I Use It?

Habari XB Client is a client component for the xmlBlaster Open Source MOM (message-oriented middleware). With this component and xmlBlaster, it is possible to exchange data between clients written in Delphi, PHP, Perl, Python, C, C++, C#, Visual Basic.net, Flash, J2ME, Java (applications, servlets, applets).

Here are some examples for software solutions built on top of a Message Broker like xmlBlaster:

- **Intranet News Ticker Application:** using the publish and subscribe communication model, news can be delivered to all registered client applications. The message sender works like a broadcast station, and does not care if clients don't listen.
- **Load Balancing:** using the point-to-point or queuing model, many 'worker' applications can be installed on different computers. Every new message sent to the queue will be delivered only to one client. The server will keep messages until they are expired or delivered to a client.
- **Persistent Storage:** messages and objects can be stored in the Object Broker and retrieved even after a restart.
- **Interprocess Communication:** applications can use point-to-point messages to exchange information between each other even if the receiver currently is not running.

xmlBlaster Features

- Messages are asynchronous accessed with the subscribe() method
- Clients receive asynchronous messages with the update() method
- Synchronous message access with the get() method
- Subscribers can use XPath expressions to filter the messages they wish to receive

About xmlBlaster

xmlBlaster is a publish/subscribe and point to point 100% Java based MOM server (message-oriented middleware) which exchanges messages between publishers and subscribers. The message is described with XML-encoded meta information. Messages may contain everything, GIF images, Java objects, Python scripts, XML data, a word document, plain text - just anything. Communication with the server is based on socket, CORBA (using JacORB), RMI, XmlRpc, HTTP or email, clients are free to choose their preferred protocol. Other protocols like SOAP may be plugged in. The xmlBlaster server is pure Java and under LGPL. Read more: <http://www.xmlblaster.org/whatis.html>

License

Habari XB Client (c) betasoft - Michael Justin 2008

This copyright applies to all source code, compiled code, documentation, graphics and auxiliary files, except those parts written by other people (which are normally copyright their authors).

GENERAL TERMS THAT APPLY TO COMPILED PROGRAMS AND REDISTRIBUTABLES

You may write and compile your own application programs using the library. You may reproduce and distribute, in executable form only, programs which you create using the library without additional license or fees, subject to all of the conditions in this statement.

The license granted in this statement for you to create your own compiled programs and distribute your programs and the Redistributables (if any) is subject to all of the following conditions: (i) all copies of the programs you create must bear a valid copyright notice, either your own or the betasoft copyright notice that appears on the Software; (ii) you may not remove or alter any betasoft copyright, trademark or other proprietary rights notice contained in any portion of betasoft libraries, source code, Redistributables or other files that bear such a notice; (iii) betasoft provides no warranty at all to any person, other than the Limited Warranty provided to the original purchaser of the Software, and you will remain solely responsible to anyone receiving your programs for support, service, upgrades, or technical or other assistance, and such recipients will have no right to contact betasoft for such services or assistance; (iv) you will indemnify and hold betasoft, its related companies and its suppliers,

harmless from and against any claims or liabilities arising out of the use, reproduction or distribution of your programs; (v) your programs must be written using a licensed, registered copy of the Software; (vi) your programs must add primary and substantial functionality, and may not be merely a set or subset of any of the libraries (including runtime libraries), code, Redistributables or other files of the Software; (vii) regardless of any modifications which you make and regardless of how you might compile, link, or package your programs, the libraries (including runtime libraries), code, Redistributables, and/or other files of the Software (including any portions thereof) may not be used in programs created by your end users (i.e., users of your programs) and may not be further redistributed by your end users; and (viii) you may not use betasoft's or any of its suppliers' names, logos, or trademarks to market your programs, except to state that your program was written using the Software.

All betasoft libraries, source code, Redistributables and other files remain betasoft's exclusive property. Regardless of any modifications that you make, you may not distribute any files (particularly betasoft source code and other non-executable files).

LIMITED WARRANTY

No warranty of any sort, expressed or implied, is provided in connection with the library, including, but not limited to, implied warranties of merchantability or fitness for a particular purpose. Any cost, loss or damage of any sort incurred owing to the malfunction or misuse of the library or the inaccuracy of the documentation or connected with the library in any other way whatsoever is solely the responsibility of the person who incurred the cost, loss or damage. Furthermore, any illegal use of the library is solely the responsibility of the person committing the illegal act. By using this program you accept these responsibilities, and give up any right to seek any damages against the authors in connection with this program.

Installation

Requirements

Development Environment

- CodeGear Delphi 2009 (or higher)
- Indy 10.5.7

Message Broker

- xmlBlaster 1.5 (or higher)

Upgrades

If you upgrade from older versions, make a backup of your existing version and make sure that you also have a backup of your own source codes.

If you upgrade from old versions, component properties may have changed and this could cause error messages when you open existing projects with the new version installed.

Architecture

THabariXBExpressBase Component

The THabariXBExpressBase component is the base class for the Habari XB connection component. It contains methods which represent the xmlBlaster methods, using the same Qos (Quality of Service) and Key objects which are documented in the xmlBlaster reference¹.

For example, opening a connection will require a ConnectQos object which has to be configured and passed to the Connect method:

```
procedure TMainForm.BtnConnectClick(Sender: TObject);
var
  ConnectQos: IConnectQos;
begin
  with HabariXBExpressBase1 do
  begin
    Host := EditIP.Text;
    ConnectQos := TConnectQos.Create('Username', 'Password');
    ConnectQos.SessionName := EditSession.Text;
    Connect(ConnectQos);
  end;
end;
```

Note

Most methods return objects which contain status informations or lists of message objects, these return values use interfaces to avoid memory leaks.

¹<http://xmlblaster.org/xmlBlaster/doc/requirements/requirement.html>

THabariXBExpress Component

The THabariXBExpress component introduces high level methods and properties. With this component, many commands can be configured by setting component properties, without the need to create instances of Qos and Key objects first.

For example, a connection can be configured and used by setting OptionsConnect Properties:

```
with MyHabariExpress do
begin
  OptionsConnect.UserName := 'SimpleReader';
  OptionsConnect.PassWord := 'secret';
  OptionsConnect.SessionName := 'SubscriberTool';
  OnUpdate := MyOnUpdate;
  Connect;
end;
```

Note

most options can be set at design time and at runtime, but not all options may be changed while a connection is open

Work In Progress

The HabariXBExpress component is still under development, future versions might introduce changes in the design of methods and properties which break existing code. Please use the low-level component HabariXBExpressBase in critical applications

Examples

Sending Messages

This example uses the THabariXBExpress component:

```
program PublisherTool;

{$APPTYPE CONSOLE}

uses
  btHabariXBExpress,
  JclSysInfo,
  BTLog,
  SysUtils;
```

JclSysInfo is part of the Jedi Code Library (JCL) and contains the GetIPAddress and GetLocalComputerName functions.

If for some reason the IP address 'localhost' or '127.0.0.1' does not work, these utility functions help to get the current IP address. Note however that these methods could fail if more than one network adapter card is installed.

The BTLog unit contains the declaration of ILogging, an interface which will be used for the logging of debug messages to console.

```
const
  NUM_MESSAGES = 10000;

var
  I: Integer;
```

The demo will publish 10000 messages.

```
begin
with THabariXBExpress.Create(nil) do
try
  Host := JclSysInfo.GetIPAddress(GetLocalComputerName);
  OptionsConnect.UserName := 'SimpleReader';
  OptionsConnect.PassWord := 'secret';
  OptionsConnect.SessionName := 'PublisherTool';
  OptionsPublish.DestinationName := 'PublisherToolTopic';
  Connect;
  for I := 0 to NUM_MESSAGES - 1 do
  begin
    if ((I+1) mod 100) = 0 then
    begin
      WriteLn(Format('Message %d', [I + 1]));
    end;
    // send the message
    try
      Publish(Format('Message %d', [I + 1]));
    except
      on E: Exception do
      begin
        WriteLn('Exception: ' + E.Message);
      end;
    end;
    end;
    Disconnect;
  finally
    Free;
  end;
  WriteLn('Press any key');
  ReadLn;
end.
```

Subscribe Messages

This example uses the THabariXBExpress component. A subclass (here called THabariExpressSubscribeDemo) of THabariXBExpress is necessary to implement the OnUpdate handler method.

Example

```
program SubscriberTool;

{$APPTYPE CONSOLE}

uses
  btHabariXBExpress,
  xbQosInterfaces,
  xbQosTypes,
  BTLog,
  JclSysInfo,
  SysUtils;

const
  NUM_MESSAGES = 1000;

var
  SubscribeKey: ISubscribeKey;
  SubscribeQos: ISubscribeQos;
```

Declaration of THabariExpressSubscribeDemo helper class with Update handler method:

```
type
  THabariExpressSubscribeDemo = class(THabariXBExpress)
  private
    procedure MyOnUpdate(const SessionID, Key, Content, Qos: string);
  end;
```

Implementation of MyOnUpdate handler, using the built-in logger to write the incoming message to the console:

```
{ THabariExpressSubscribeDemo }

procedure THabariExpressSubscribeDemo.MyOnUpdate(const SessionID, Key,
Content,
  Qos: string);
begin
  Logger.Info(Content);
end;
```

Main program code. Assigns the OnUpdate handler, Connects and receives messages for up to 10 seconds.

```
begin
  with THabariExpressSubscribeDemo.Create(nil) do
  try
    Host := JclSysInfo.GetIPAddress(GetLocalComputerName);
    UserName := 'SimpleReader';
    PassWord := 'secret';
    OnUpdate := MyOnUpdate;
    Connect;
    SubscribeKey := TSubscribeKey.Create;
    SubscribeKey.Oid := 'PublisherToolTopic';
    SubscribeQos := TSubscribeQos.Create;
    Subscribe(SubscribeKey, SubscribeQos);
    Sleep(10000);
    Disconnect;
  finally
    Free;
  end;
  WriteLn('Press any key');
  ReadLn;
end.
```

HabariXBExpress

Events

OnPing

Event handler which will be called when the Server sends a ping message to the client.

OnUpdate

Event handler which will be called when the Server sends a message to the client.

Methods

Connect

Connect to the xmlBlaster message broker using the configuration in OptionsConnect.

Disconnect

Disconnect using the configuration in OptionsDisonnect.

Erase

The Erase method allows to erase messages in xmlBlaster.

Publish

Publish a text message using the configuration in OptionsPublish.

Subscribe

Subscribe to messages.

Unsubscribe

The UnSubscribe method allows to cancel message subscriptions in xmlBlaster. To unsubscribe you pass the subscriptionId you got from subscribe.

Properties

Active

The Active property may be used instead of the Connect and Disconnect methods.

Host

The xmlBlaster server IP address.

OptionsConnect

AllowPeerToPeer	This allows to suppress receiving PtP messages. Default: True
ClearSessions	For administrative purposes we can set OptionsConnect.ClearSessions := True which will destroy all other login sessions using the current session name in SessionName. Default: False
MaxSessions	How often the same client may login. Default: 10
Password	Password Default: empty string
Persistent	Once you have made a session persistent and you are reconnecting, the new connection will ignore the persistent flag. Once made persistent, the session will remain persistent. Default: False
SessionName	The session name. A default value will be created which can be overriden while the connection is closed.
SessionTimeOut	Timeout until session expires if no communication happens. Default: 86400000 msec (one day)
Username	Username Default: empty string

OptionsDisconnect

ClearSessions	If we shall kill all other sessions of this user on logout Default: False
DeleteSubjectQueue	If subject queue shall be deleted with last user session logout. Default: False

OptionsErase

ForceDestroy	Kill a topic even if there are pending updates or subscriptions Default: False
HistoryNumEntries	Default is to erase the current entry (numEntries='1'), '-1' erase all Default: 1
OID	The Destination name
QueryType	qtEXACT or qtXPATH Default: qtEXACT
QueryString	XPATH query string

OptionsGet

HistoryNumEntries	Default is to deliver the current entry (numEntries='1'), '-1' deliver all. Default: 1
HistoryNewestFirst	HistoryNewestFirst let you change the delivery order. Default: True
OID	The destination (topic) name.
QueryType	qtEXACT or qtXPATH Default: qtEXACT
QueryString	XPATH query string
WantContent	If false, the update contains not the content (it is a notify of change only) Default: True

OptionsPublish

ContentMime	A MIME type like "image/gif". Default: 'text/plain'
OID	The destination (topic) name.
ForceDestroy	Control message life cycle on message expiry. Default: False.
ForceUpdate	Send message to subscriber even if the content is the same as the previous? Default: True
LifeTime	Control the life time of a message, given in milliseconds. Passing -1 milliseconds asks the server for unlimited lifespan, which the server may or may not grant (but currently does grant with the

	default configuration). Default: -1
Persistent	Mark a message to be persistent. Default: False
Priority	Set message priority value, 5 is default.
Subscribable	As a default setting you can subscribe on all messages (PtP or PubSub). Set it to False to make PtP message invisible for subscribes. Default: True
Volatile	Mark a message to be volatile or not. A non-volatile messages stays in memory as long as the server runs or the message expires. A volatile messages exists only during publish and processing it (doing the updates). Defaults to false. NOTE: This is a convenience method for LifeTime := 0 and ForceDestroy := False Default: False

OptionsSubscribe

HistoryNumEntries	Default is to deliver the current entry (numEntries='1'), '-1' deliver all. Default: 1
HistoryNewestFirst	HistoryNewestFirst let you change the delivery order. Default: True
Meta	Don't send me the xmlKey meta data on updates. Default: True
MultiSubscribe	Ignore a second subscribe on same oid or XPATH.
Notify	Suppress erase event to subscribers. Default: True
OID	The Destination name
QueryType	qtEXACT or qtXPATH Default: qtEXACT
QueryString	XPATH query string
Persistent	You can mark a subscription to be persistent. If the server crashes or is restarted the clients session and this subscription is restored.
SubscriptionID	Force a subscription ID from client side.
UpdateOneway	Send callbacks messages for this subscription with the better performing updateOneway() instead of the more reliable update() (default: false)

WantContent If false, the update contains not the content (it is a notify of change only)
Default: True

WantInitialUpdate Do we want to have an initial update on subscribe if the message exists already?
Default: True

WantLocal Allow or inhibit the delivery of messages to myself if i have published it.

OptionsUnSubscribe

Port

The server port. Default: 7607

Known Limitations

Message Exchange

The updateOneway method is not supported.

Object Exchange

Object exchange is not supported, however it is possible using third party libraries like tkJSON or SuperObject.

References

Message Broker

xmlBlaster <http://xmlBlaster.org/>

IDE

CodeGear Delphi <http://www.codegear.com/delphi>

Communication Libraries

Indy 10 <http://www.indyproject.org/>

JSON Libraries

SuperObject <http://www.progdigy.com>

IkJSON <http://sourceforge.net/projects/ikjson>

Release Notes

Version 1.2

xmlBlaster 2.0.0 this version has been tested with xmlBlaster 2.0.0
Indy tested with 10.5.7 revision 3865
JCL tested with jcl-1.105.1.3400

Version 1.1

Released April 14, 2009

xmlBlaster 1.6.4 this version has been tested with xmlBlaster 1.6.4
JCL upgraded to JCL 1.105
doxygen upgraded to doxygen 1.5.8
Delphi 2009 fixed compiler warnings
Renamed units renamed core unit file names, they start with xb, component unit names start with bt

Version 1.0

Released May 29, 2008

Index

Reference