jBASE

jBASE jRCS

What is jRCS?

- Lightweight remote connector for jBASE
- Provides access to jBC-like functionality from GUI and web applications
- Adaptable to multiple platforms

Similar Products

- PickODBC for Raining Data D3
- UniObjects for IBM’s U2 suite

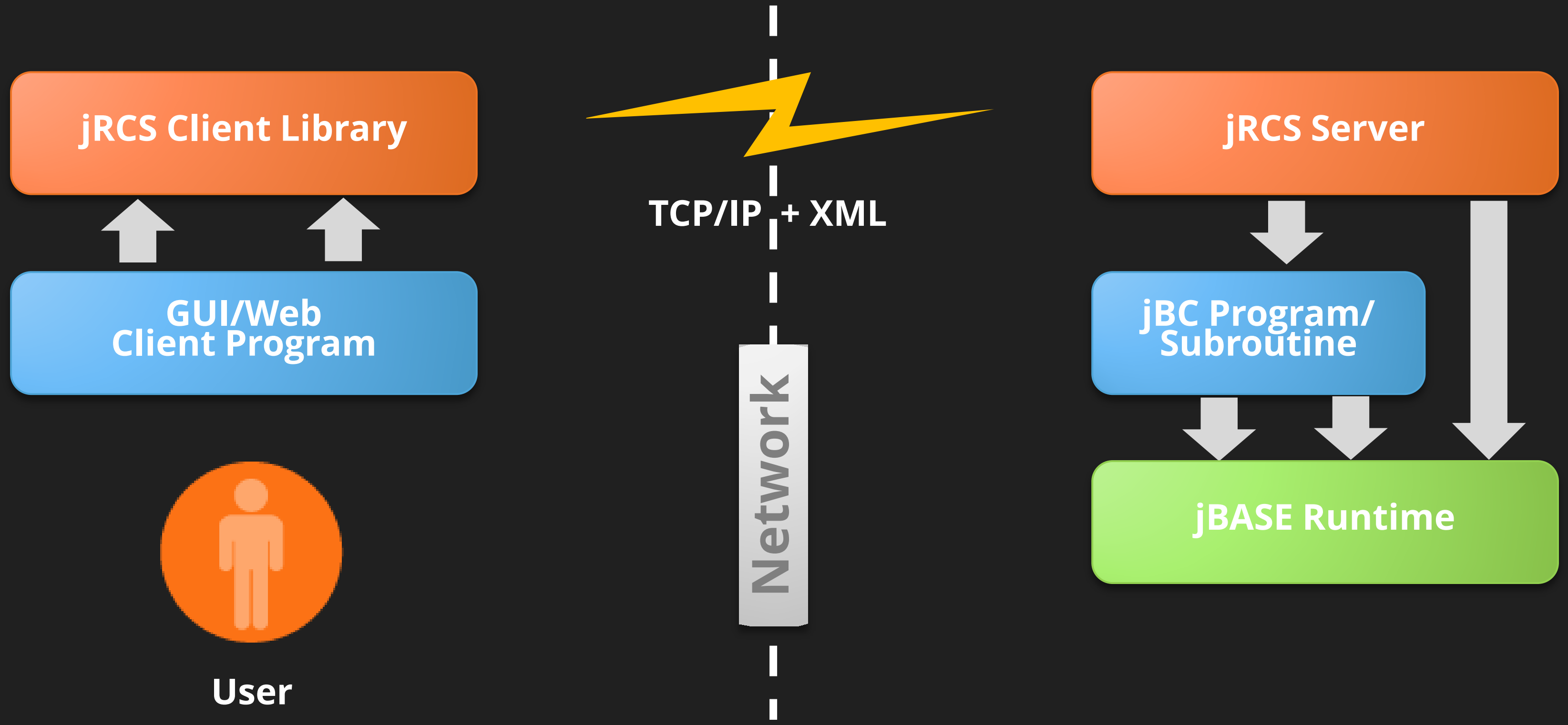
Why jRCS?

- Increased demand for GUI and web-based multi-tier applications
- Improved end user experience
- Ability to retain most of the legacy jBC code base
- Smooth transition to the world of Windows and other graphical environments

jRCS Features

- Small and fast remote connector
- Leverages established technologies
  - TCP/IP
  - XML
- Easy portability
- Open protocol

jRCS Architecture



jRCS Capabilities

- Connection establishment and termination
- User authentication via underlying OS
- jBC environment setup
- Calling jBC programs and subroutines

jRCS Capabilities (Cont’d)

- Performing jQL-like conversions
- Opening, reading and writing jBASE files
- Full locking support
- Creating select lists from files

jRCS Capabilities (Cont'd)

- Selecting records using jQL
- Selecting jBASE indexes
- Managing common variable blocks
- Fast client-side dynamic array support (jBASE-supplied client libraries only)

Supported Server Platforms

- Windows 2000, XP and Server 2003
- 32-bit AIX (64-bit server also available)
- RedHat Enterprise Linux

Available Client Libraries

- C/C++ client (low-level integration)
- ActiveX client for VB 6.0 (Windows only)
- Microsoft .NET client (Windows only)
  - Framework 1.1 and 2.0 supported
- Java client for JDK 1.4.2 and newer
- Microsoft .NET Compact Framework client

Using jRCS

jRCS Authentication

- Underlying OS user names and passwords are used for authentication
- Permissions are set up based on user credentials supplied at logon
- User is placed in his/her home directory

Environment Setup

- Environment is set up based on user’s jRCS resource file
  - On Unix: $HOME/.jrcsrc
  - On Windows: %HOME%\jrcsrc.cmd
- All jBC environment variables can be set

jrcsrc.cmd on Windows

- Follows the cmd.exe syntax for environment variables
- Percent sign substitutions can be used
- Programs may not be executed
- Example:

```
Set JBCOBJECTLIST = %HOME%\lib
Set JEDIFILEPATH = %HOME%
```

.jrcsrc on Unix

- Follows the syntax of the Bourne shell (sh)
- Dollar sign substitutions may be used
- Program execution and backquote substitution is not permitted
- Example:

```
JBCOBJECTLIST = $HOME/lib
export JBCOBJECTLIST
JEDIFILEPATH = $HOME
export JEDIFILEPATH
```

jRCS .NET Client – Programmer’s View

Typical jRCS Session

- Establish a connection and authenticate
- Run business logic
  - Call a subroutine
  - Execute a program
  - Open and read or write a file
  - Generate a select list
- Terminate the connection

Connection Establishment

- Create a JConnection object
- Call the Open method and pass the user name, password and host name
- Example:

```
Dim _conn As New JConnection
_conn.Open(“localhost”, JConnection.JRCS_PORT, “test”, “test”, “”)
```

File Management

- Use OpenFile method in JConnection to create a JFile object
- Read and write records using JFile methods
- Example:

```
Dim _file As JFile = _conn.OpenFile(“CUSTOMER”)
Dim _record As JDynArray = _file.Read(“12345”, False, False)
_record.Replace(“New Customer Name”, 1)
_file.Write(“12345”, _record, False)
```

Dynamic Arrays

- Create a JDynArray object
- Use its methods to extract, replace and insert data, locate fields, count attributes
- Example:

```
Dim _array As New JDynArray
_array.Insert(“Field 1”, 1)
_array.Replace(“New Field 1”, 1)
Debug.WriteLine(_array.Extract(1))
Debug.WriteLine(“Number of attributes: “ & _array.DCount(_array.AM))
```

Select List Manipulation

- Use Select or SSelect in a JFile object to create a JSelectList
- Use Execute method of JConnection to return a JSelectList
- Use For Each … Next or ReadNext method to iterate through the list

```
Dim _file As JFile = _conn.OpenFile(“CUSTOMER”)
Dim _list As JSelectList = _file.SSelect
For Each _key As String In _list
        Debug.WriteLine(_key)
Next
```

Calling Subroutines

- Use Call method in JConnection to call a subroutine
- Parameters are passed as an array of strings or JDynArray objects
- Example:

```
Dim _parameter As New JDynArray(“This will be passed and returned”)
Dim _parms() As JDynArray = New JDynArray() { _parameter }
_conn.Call(“MY_SUBROUTINE”, _parms)
Debug.WriteLine(“Returned value: “ & _parms(0))
```

Executing Programs

- Use Execute or ExecuteAndStore method in JConnection
- ExecuteAndStore allows captured output to be read block-by-block
- Select lists may be passed and returned
- Example:

```
Dim _execResults As JExecuteResults = _
        _conn.Execute(“LIST CUSTOMER”, _
        JExecFlags.EXEC_GET_CAPTURE, Nothing)
Debug.WriteLine(“Captured text: “ & _execResults.CaptureString)
```

Error Handling

- Objects of class JException are thrown back
- Use the Message property of JException to get the error message
- Example:

```
Try
        Dim _record As JDynArray = _file.Read(“12345”, _
                False, False)
Catch _exception As JException
        Debug.WriteLine(“Error: “ & _exception.Message)
End Try
```