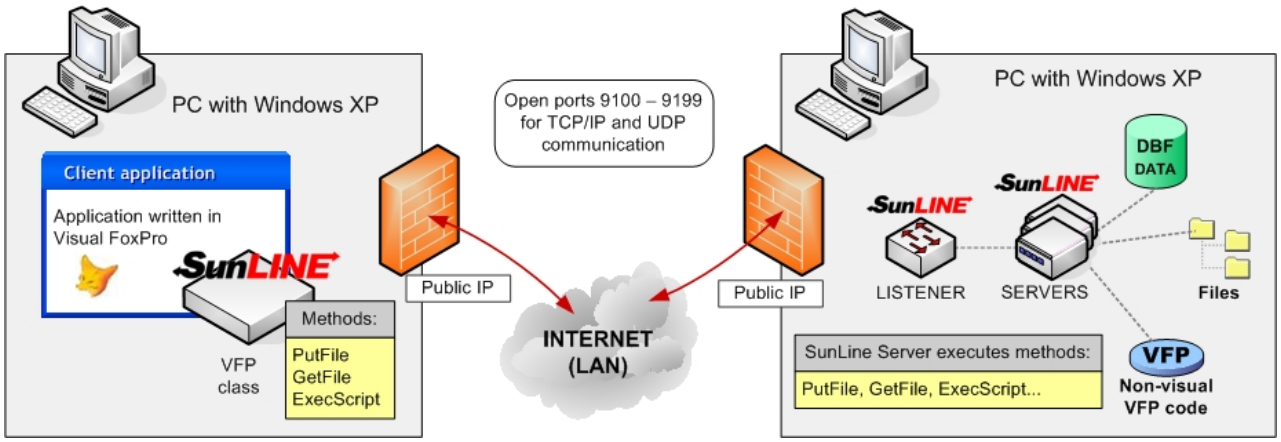
SunLINE SDK for VFP

SunSoft SunLINE SDK is a set of VFP libraries designed for software developers. It allows development of client/server solutions using TCP/IP protocol without need of special expensive application or database servers and clients.

SunLine SDK for VFP provides both application VFP server and client VFP APP components for integrating into developer solutions. These components allow connect local computers to remote computers over LAN or internet using TCP/IP protocol with VFP commands executed on the remote computer. The client PC can get access to all local DBF data and sources of the remote computer using VFP commands in PRG scripts. One client application can work with more server stations.



SunLINE SDK key features

- With SunLINE you can create simple application server for work with DBF data and for access to local sources of remote computer.
- Application server SunLINE is driven by VFP PRG scripts, which are written on the client side of application – thus the application logic is not stored on the server but in the client application code. By this process, developer can expand its functionality without modification of server code.
- All components of SunLINE SDK have been written in Visual FoxPro 9.0. SunLINE client scripts are based on VFP 9 language and allow using all non-visual VFP 9 commands (basically it is standard PRG). Thus can client application effectively manipulate with DBF data and other local data and functions on the server side.
- Client and server are interconvertible (dual) systems – you can install client and server parts of SunLINE on both computers. This allows using extended client/server/client functionality; systems are changing roles according to current requirement.
- In SunLINE SDK Full version is the communication between the client and server secured by a standard stream cipher RC4 (RSA Data Security). This cipher is commonly used in encrypted communication and it is almost impossible to break it.

SunLINE SDK modules

SunLINE Client

VFP APP class library, you can include it to any VFP application. Application with integrated SunLINE client runs on client PC. Library contains methods for creating server scripts and for file transfers.

SunLINE Server

VFP executable file running in hidden mode on the server PC. Basically it represents universal application server driven by VFP commands and scripts from the client PC.

SunLINE Listener

VFP executable file running in hidden mode on server PC (it is shown only in SysTray). It provides communication between the client and server components.

Installation

Start the installation by executing SETUP_SUNLINE_LITE.EXE. It will install both client and server sides. All components will be default installed to the C:\Program Files\SunSoft\SunLINE SDK directory. Listener represents the server side and can be automatically run on Windows startup. Client side represents a VFP library used in the client application.

SunLINE client

SunLINE client initialization

For proper function of SunLINE client you need initialize one PUBLIC object variable in the Microsoft Visual FoxPro 9.0 development environment. The variable can have any name; it will be called in this documentation **oSunLine**.

Example of SunLINE client initialization

PUBLIC oSunLine

oSunLine = NEWOBJECT('sunline', 'sunline.vcx', 'C:\Program Files\SunSoft\SunLINE SDK Lite\Libs\SunLINE.app')

- 1. parameter: sunline – required keyword
- 2. parameter: sunline.vcx – required keyword
- 3. parameter: full path to SunLINE.app file (for VFP 8 use SunLINE8.app)

After initialization exists **oSunLine** object with all its properties and methods required by SunLINE client.

Establishing connection, address assignment

oSunLine.RemoteHost

Target address for SunLINE client connection is defined by setting properties of **oSunLine.RemoteHost** object.

Properties:

oSunLine.RemoteHost.IP = '192.168.1.155'

oSunLine.RemoteHost.Port = 9100

oSunLine.RemoteHost.Info = ''

oSunLine.RemoteHost.InfoAddress = ''

- Property **.IP**: IP address of the PC in the internet or LAN where is running SunLINE Listener
- Property **.Port**: number of TCP/IP protocol port where is listening SunLINE Listener
- Property **.Info** (optional): information about SunLINE client. This information is shown in the SunLINE Listener window.
- Property **.InfoAddress** (optional): information about the place where is running SunLINE client. This information is shown in the SunLINE Listener window.

oSunLine.StartClient()

oSunLine.StartClient() method checks all necessary setting and initializes other objects required for proper SunLINE client functionality.

Parameters:
This method has no parameters.

Return value:
Logical data type (True , False) – according to successful test of the **oSunLine.RemoteHost** object settings.

oSunLine.Connect()
oSunLine.Connect() establishes the connection between the client and SunLINE Listener.

Parameters:
This method has no parameters.

Return value:
Logical data type. Returns (.T.) if the connection has been established; otherwise returns (.F.).

Terminating connection, releasing SunLINE client

oSunLine.Disconnect()
oSunLine.Disconnect() method terminates connection with SunLINE Listener. However all properties and child objects of SunLINE client will remain in memory. The connection can be established again by calling **oSunLine.Connect()** method.

Parameters:
This method has no parameters.

Return value:
Logical data type. Returns (.T.) if the connection has been terminated; otherwise returns (.F.). If there was no active connection, the return value is (.F.).

oSunLine.StopClient()
oSunLine.StopClient() method terminates connection with SunLINE Listener and initializes all SunLINE client objects. It is not necessary call before this method the **oSunLine.Disconnect()** method. You can continue to work with SunLINE client by calling **oSunLine.StartClient()** method.

Parameters:
This method has no parameters.

Return value:
Logical data type. Returns (.T.) if SunLINE client has been stopped; otherwise returns (.F.).

oSunLine.End()
oSunLine.End() method definitively terminates SunLINE client. All objects will be released from the memory and the temporary files will be deleted. After calling this method you cannot continue to work with the SunLINE client.

Parameters:
This method has no parameters.

Return value:
This method has no return value.

Connection test between SunLINE client and SunLINE Listener

oSunLine.IsConnected()

oSunLine.IsConnected() method tests if is established the connection between the client and SunLINE Listener.

Parameters:
This method has no parameters.

Return value:
Logical data type. Returns (.T.) if the connection is active (live); otherwise returns (.F.).

Commands execution

oSunLine.ExecRemoteScript()

oSunLine.ExecRemoteScript() method sends any non-visual Microsoft Visual FoxPro command to the remote SunLINE server for execution.

Parameters:
This method has 1 parameter: VFP command as string.

Return value:
Return value is a string representing the result of the remote VFP command execution. If the command has no result to return and the command has been successfully executed, the return value is „.T.". If the command execution failed, the return value is „Error".

Example:
cDateTime = **oSunLine.ExecRemoteScript**('RETURN DATETIME()')
Return: cDateTime = 31.08.2006 13:27:54
cDateTime = **oSunLine.ExecRemoteScript**('RETURN DATETYME()')
Return: cDateTime = Error
cResult = **oSunLine.ExecRemoteScript**('cDate = DATETIME()')
Return: .T.

LITE version limitations:
You can call during one instance of SunLINE the method **oSunLine.ExecRemoteScript()** 10 times. Then is necessary reset the SunLINE object.

oSunLine.ExecRemoteScriptFile()

oSunLine.ExecRemoteScriptFile() method sends a program file of any non-visual Microsoft Visual FoxPro commands to the remote SunLINE server for executionn. The size of the PRG file (script) is not limited and it can contain any non-visual VFP source code. In the script can be opened tables, executed various calculations and system operations as well.

Parameters:
This method has 6 parameters.

- 1. parameter: full path to the PRG file (script) on the client side. This file is transferred to the remote side and executed by the server.
- 2. – 6. parameter (optional): Parameters used as parameters for the PRG file.

Return value:
Return value is a string representing the result of the PRG file execution. It depends only on the developer what data will the routine return.

Example:
cResult = **oSunLine.ExecRemoteScriptFile**('c:\prog\sample.prg', 'AB')
1. parameter: full path to PRG file
2. parameter: String AB is passed as parameter for the PRG
Return: cResult = (value of any type)

LITE version limitations:
PRG file is limited to a maximum of 20 lines of code.

File transfer:

oSunLine.GetFile()

oSunLine.GetFile() method copies any file from the remote side to the local client side.

Parameters:
This method has 2 parameters.

- 1. parameter: Full path to the source (remote) file
- 2. parameter: Full path to the target (local) file

Return value:
Logical data type. Returns (.T.) if the transfer succeeded; otherwise returns (.F.).

Example:
oSunLine.GetFile('c:\files\source.dbf', 'c:\files\target.dbf')
Remote file: c:\files\source.dbf is transferred to the client side.

oSunLine.PutFile()

oSunLine.PutFile()method copies any file from the client side to the remote (server) side.

This method has 2 parameters.

- 1. parameter: Full path to the source (local) file
- 2. parameter: Full path to the target (remote) file

Return value:
Logical data type. Returns (.T.) if the transfer succeeded; otherwise returns (.F.).

Example:
oSunLine.PutFile('c:\files\source.dbf', 'c:\files\target.dbf')
Local file: c:\files\source.dbf is transferred to the remote side.

Operation timeout:

oSunLine.TimeOut

oSunLine.TimeOut property represents timeout value for the execution of the SunLINE client operations. For each operation can be set a different timeout limit.

Properties:
oSunLine.TimeOut = 5000

- 5000: timeout limit in milliseconds

Example:
oSunLine.TimeOut = 5000
cResult = **oSunLine.Connect**()
SunLINE client attempts to connect to the SunLINE listener for period of 5 seconds. If it during this period fails to connect, the oSunLine.Connect() method automatically returns error.

Default value:
The default value of the **oSunLine.TimeOut** property is 0 – method attempts to execute the operation indefinitely.

SunLINE client properties:

oSunLine.oLang.LngFile

oSunLine.oLang.LngFile represents full path to the language file of the SunLINE client. If is this property set and proper language file exists, all SunLINE texts will be displayed in the selected language.

Example:

oSunLine.oLang.LngFile = ' C:\Program Files\SunSoft\SunLINE SDK Lite\Lang\english.lng'

oSunLine.ConnectResult

oSunLine.ConnectResult property represents the result of the **oSunLine.Connect**() method. This property has the same value as the result of the **oSunLine.Connect**() method and is intended for later use.

oSunLine.GetFileResult

oSunLine.GetFileResult property represents the result of the **oSunLine.GetFile**() method. This property has the same value as the result of the **oSunLine.GetFile**() method.

oSunLine.PutFileResult

oSunLine.PutFileResult property represents the result of the **oSunLine.PutFile**() method. This property has the same value as the result of the **oSunLine.PutFile**() method.

oSunLine.RemoteScriptResult

oSunLine.RemoteScriptResult property represents the result of the **oSunLine.ExecRemoteScript**() or **oSunLine.ExecRemoteScriptFile**() method. This property has the same value as the result of the **oSunLine.ExecRemoteScript**() and **oSunLine.ExecRemoteScriptFile**() methods.

oSunLine.RemoteScriptBody

oSunLine.RemoteScriptBody property contains the command or the body of the script file sent to the remote server for execution. It represents the commands or script files sent by the **oSunLine.ExecRemoteScript**() or **oSunLine.ExecRemoteScriptFile**() methods.

oSunLine.RemoteScriptError

oSunLine.RemoteScriptError is object that contains properties representing the script execution error. If methods **oSunLine.ExecRemoteScript**() or **oSunLine.ExecRemoteScriptFile**() return „Error", property **oSunLine.RemoteScriptError** is filled with details about the error.

Properties:

oSunLine.RemoteScriptError.Code – Microsoft Visual FoxPro error code

oSunLine.RemoteScriptError.Message – VFP error message

oSunLine.RemoteScriptError.Line – line of the PRG file where the error occurs

oSunLine.RemoteScriptError.Procedure – procedure in the PRG file where the error occurs

oSunLine.RemoteScriptError.Details – error details

oSunLine.LastError

oSunLine.LastError contains information about the last error occurred during the script execution. In opposite of the **oSunLine.RemoteScriptError** property, this property is not cleared before every script execution.

SunLINE Listener

SunLINE Listener is executable application written in Microsoft Visual FoxPro 9. After startup it is set do listening mode and is waiting for incoming connection from SunLINE clients. SunLINE listener has some parameters stored in the SunLINE.ini file and you can change it by editing the file.

SunLINE.ini properties:
Port=9100 – number of TCP/IP protocol port where is SunLINE Listener awaiting incoming connection from clients
Language=[Lang\english.lng] – full path to the SunLINE Listener language file

SunLINE window displays information about active connections with SunLINE clients. The second tab displays the connection history. It is possible from this window close appropriate connection.

LITE version limitations:
SunLINE Listener allows establishing only one client - server connection.

SunLINE Server

SunLINE Server is executable application written in Microsoft Visual FoxPro 9. It represents VFP application server which executes commands and scripts sent by the client. The server is activated by the SunLINE Listener on the client demand and runs in hidden mode.

System requirements

- Client requires Microsoft Visual FoxPro 9 (8 SP1) development environment; server requires VFP 9 runtime
- Windows XP
- Permanent connection to internet with public IP addresses and open ports 9100 – 9199

ATTENTION, in this case it is necessary to secure connected computers by firewalls, eventually use VPN.