

TSMInetBase component

[See also](#) [Properties](#) [Methods](#) [Events](#)

Unit

[SMInBase](#)

Description

This component is a basic socket component for SMInternet suite.

The TSMWhols, TSMInetDateTime and TSMUPSTracker components are inherited from TSMInetBase type.

About property

Applies to

[TSMInetBase](#) component

Declaration

property About: TSMInetAbout;

Description

This is design-time property - in Object Inspector you can read a basic information about SMInternet suite.

Address property

Applies to

[TSMInetBase](#) component

Declaration

property Address: **string**;

Description

Address is the IP address of the server system.

Address is a string of four numeric (byte) values in the standard internet dot notation, such as

123.197.1.2

For client sockets, set Address to the IP address of the server to which the socket should connect. When the connection is opened, the value of Address is bound to the connection. If the socket specifies the Host property, the address for the connection is taken from the IP address associated with Host, rather than the value of Address. Specifying the server system by giving the IP address is faster, because the socket doesn't need to search for the IP address associated with the host name before it can locate the server system.

A single system may support more than one IP address.

Note

Trying to change Address when the connection is open will raise an ESocketError exception.

ClientType property

Applies to

[TSMInetBase](#) component

Declaration

property ClientType: TClientType;

Description

ClientType specifies whether the client socket reads and writes information asynchronously over the socket connection.

Set ClientType to ctNonBlocking to enable the client socket to respond to asynchronous reading and writing events. When ClientType is ctNonBlocking, execution is not blocked by reading and writing over the socket connection. OnRead or OnWrite events occur when the socket needs to read or write over the connection.

Set ClientType to ctBlocking to force all reading and writing to occur synchronously. It is a good idea to include the client socket object in a thread if the ClientType is ctBlocking, so that I/O does not block all execution within the client application.

When ClientType is ctBlocking, use a TWinSocketStream object for reading and writing. TWinSocketStream prevents the application from hanging indefinitely if a problem occurs while reading or writing. It also can wait for the socket connection to indicate its readiness for reading.

Use a non-blocking socket when the socket needs to synchronize reading and writing with server sockets.

Host property

Applies to

[TSMInetBase](#) component

Declaration

```
property Host: string;
```

Description

Host is an alias for the IP address of the server system.

Host is a string containing the domain name and service of a particular system, such as <http://www.scalabium.com>

For client sockets, set Host to the system with which the client socket should form a connection. When the socket opens a connection, it looks up the IP address for the server socket using the value of Host.

Some servers change the system or IP address that is associated with a particular host name. Using a host name allows the client socket to find the abstract site represented by the host name, even when it has moved to a new IP address.

If Host is set, it takes precedence over the Address property when looking up the address of the server.

Note

Trying to change Host when the connection is open will raise an ESocketError exception.

KeepConnection property

[Example](#)

Applies to

[TSMInetBase](#) component

Declaration

property KeepConnection: Boolean;

Description

When KeepConnection is True, after every request processing the connection will be still active. Else component automatically call the Disconnect method.

So when you'll process a batch of request to Internet, set a KeepConnection in True and it will increase a performace because component will not connect/disconnect after every process.

Port property

Applies to

[TSMInetBase](#) component

Declaration

property Port: Integer;

Description

Port is the ID number that identifies the server socket connection.

Port numbers allow a single system, identified by the Host or Address property, to host multiple connections simultaneously. Many values of Port are associated by convention with a particular service such as ftp or http.

For server sockets, Port is the ID of the connection on which the server socket listens for client requests. Server sockets generally set Port to a predefined value which clients can use to initiate connections.

For client sockets, Port is the ID of the desired server connection. The value of Port is usually associated with the service the client wishes to use on the server application.

Note

Trying to change Port when the connection is open will raise an ESocketError exception.

Execute method

[See also Example](#)

Applies to

[TSMInetBase](#) component

Declaration

```
function Execute: string;
```

Description

This method will activate and process the request to server. Result of this function is a result which is retrieved from server.

For example, you can define every property of TSMWhols (domain for checking, server list etc) and call an Execute for processing.

GetQuery method

[See also Example](#)

Applies to

[TSMInetBase](#) component

Declaration

```
function GetQuery: string; virtual;
```

Description

This is a virtual method which must return a string which will be sent to server as query request. Will be called automatically as part of Execute method...

Every inherited component must override this method for own implementation.

For example, TSMWhols must send a domain name with trailing CRLF, but TSMUPSTracker must send the GET-request by HTTP protocol with formatted string from special parameters (tracking number, tracking type etc

Connect method

[See also](#)

Applies to

[TSMInetBase](#) component

Declaration

procedure Connect;

Description

Connect opens the socket connection.

Call Connect to initiate the socket connection. This method locates and connects to a server.

Disconnect method

[See also](#)

Applies to

[TSMInetBase](#) component

Declaration

```
procedure Disconnect;
```

Description

Disconnect shuts down the socket connection.

Call Disconnect to shut down the socket connection. This method terminates the connection to the server.

OnAfterExecute event

[See also](#)

Applies to

[TSMInetBase](#) component

Declaration

property OnAfterExecute: TSMIReceivedEvent;

Description

This event will be executed automatically in end of processing of Execute method. Using this handler you can activate some own actions which must be executed after process.

The Sender parameter is an internet component which finished a data processing.

OnBeforeExecute event

[See also Example](#)

Applies to

[TSMInetBase](#) component

Declaration

property OnBeforeExecute: TNotifyEvent;

Description

This event will be called automatically before any data processing (in start of Execute method).

Using this handler you can activate some own actions which must be executed before process.

The Sender parameter is an internet component which will start a data processing.

OnConnect event

[See also Example](#)

Applies to

[TSMInetBase](#) component

Declaration

property OnConnect: TSocketNotifyEvent;

Description

OnConnect occurs on client sockets just after the connection to the server is opened.

Write an OnConnect event handler for a client socket to take specific action after the connection to a server socket has been established. Depending on the service, this may be the point when the socket should start reading or writing over the connection.

When a client socket opens a connection, the following events occur:

- 1 An OnLookup event occurs prior to an attempt to locate the server socket.
- 2 The Windows socket is set up, and initialized for event notification.
- 3 An OnConnecting event occurs after the server socket is located.
- 4 The connection request is accepted by the server and completed by the client socket.
- 5 An OnConnect event occurs after the connection is established.

The Sender parameter is the client socket component that initiates the connection. The Socket parameter is the TCustomWinSocket object that describes the client endpoint of the newly formed connection.

OnConnecting event

[See also Example](#)

Applies to

[TSMInetBase](#) component

Declaration

property OnConnecting: TSocketNotifyEvent;

Description

OnConnecting occurs for a client socket after the server socket has been located, but before the connection is established.

Write an OnConnecting event handler for a client socket to take specific action just before the connection to a server socket is established. This is the first opportunity to obtain the actual port and IP address of the server endpoint of the connection that is about to form. When a client socket opens a connection, the following events occur:

- 1 An OnLookup event occurs prior to an attempt to locate the server socket.
- 2 The Windows socket is set up, and initialized for event notification.
- 3 An OnConnecting event occurs after the server socket is located.
- 4 The connection request is accepted by the server and completed by the client socket.
- 5 An OnConnect event occurs after the connection is established.

The Sender parameter is the client socket component that initiates the connection. The Socket parameter is the TCustomWinSocket object that describes the server endpoint of the connection that is about to form.

OnDisconnect event

[See also Example](#)

Applies to

[TSMInetBase](#) component

Declaration

property OnDisconnect: TSocketNotifyEvent;

Description

OnDisconnect occurs just before a client socket closes the connection to a server socket.

Write an OnDisconnect event handler to take specific action when the connection to a server socket is about to be terminated. OnDisconnect occurs after the value of the Active property is set to False, but before the actual connection is closed.

The Sender parameter is the client socket component. The Socket parameter is the TCustomWinSocket object that describes the client endpoint of the connection that is about to end.

OnError event

[Example](#)

Applies to

[TSMInetBase](#) component

Declaration

property OnError: TSocketErrorEvent;

Description

OnError occurs when the socket fails in making, using, or shutting down a connection.

Write an OnError event handler to respond to errors that arise with the socket connection. Set the ErrorCode parameter to 0 if the OnError event handler successfully deals with the error condition, to prevent an ESocketError from being raised.

The Sender parameter is the socket component. The Socket parameter is the TCustomWinSocket object that describes the local endpoint of the connection that gave rise to the error. The ErrorEvent parameter indicates what Socket was attempting to do when the error occurred. The ErrorCode parameter is the error code returned by the Windows socket API call.

OnLookup event

Applies to

[TSMLnetBase](#) component

Declaration

property OnLookup: TSocketNotifyEvent;

Description

OnLookup occurs when a client socket is about to look up the server socket with which it wants to connect.

Write an OnLookup event handler for a client socket to take specific action just before attempting to locate a server socket. This is the first opportunity to make Windows API calls that affect the client properties of the socket, such as specifying a particular port number. Use the SocketHandle property of the Socket parameter for Windows API calls.

When a client socket opens a connection, the following events occur:

- 1 An OnLookup event occurs prior to an attempt to locate the server socket.
- 2 The Windows socket is set up, and initialized for event notification.
- 3 An OnConnecting event occurs after the server socket is located.
- 4 The connection request is accepted by the server and completed by the client socket.
- 5 An OnConnect event occurs after the connection is established.

The Sender parameter is the client socket component that initiates the connection. The Socket parameter is the TCustomWinSocket object that describes the client endpoint of the connection that will be formed with the server.

Note

Changing the Address, Host, Port, or Service properties of the Sender parameter in an OnLookup event handler will have no effect on the address or port used to locate a server socket. These properties must be correct before the Open method is called.

OnReceiveTextFromSocket event

Applies to

[TSMInetBase](#) component

Declaration

property OnReceiveTextFromSocket: TSocketNotifyEvent;

Description

OnReceiveTextFromSocket occurs when a client socket should read information from the socket connection.

Write an event handler to read from the socket connection. The Sender parameter is the client socket component that initiated the connection. The Socket parameter is the TCustomWinSocket object that describes the client endpoint of the socket connection. If the socket is a blocking socket, use a TWinSocketStream object to read from the connection. Otherwise, use the methods of the Socket parameter to perform the actual reading.

OnSendTextToSocket event

Applies to

[TSMLnetBase](#) component

Declaration

property OnSendTextToSocket: TSocketNotifyEvent;

Description

OnSendTextToSocket occurs when a client socket should write information to the socket connection.

Write an event handler to write to the socket connection. The Sender parameter is the client socket component that initiated the connection. The Socket parameter is the TCustomWinSocket object that describes the client endpoint of the socket connection. If the socket is a blocking socket, use a TWinSocketStream object to write to the connection. Otherwise, use the methods of the Socket parameter to perform the actual writing.



TSMWhoIs component

[See also](#) [Properties](#) [Methods](#)

Unit

[SMWhoIs](#)

Description

The Whois client component enables the application to request information from a server about a specific domain or user. This control would be most commonly used to query the Whois server at rs.internic.net to obtain information about a specific Internet domain name or an administrative contact at that domain.

Note that local WhoIs-servers are supported for any domain extension (.de, .fr, .ru, .uk etc)

AutoSelectServer property

[See also Example](#)

Applies to

[TSMWhols](#) component

Declaration

property AutoSelectServer: Boolean;

Description

When AutoSelectServer is a True, the required server for domain check will be selected automatically from list of available servers (see ServerList property).

When AutoSelectServer is a False, component will try to retrieve an information from connected server (see Host property).

Domain property

See also [Example](#)

Applies to

[TSMWhols](#) component

Declaration

```
property Domain: string;
```

Description

This property is a name of domain which you want to check on server. For this value you'll retrieve any registered information from whois-server.

ServerList property

[See also](#)

Applies to

[TSMWhols](#) component

Declaration

property ServerList: TStrings;

Description

Here you can add/delete/edit a list of available whois-servers which will be used for domain name checking.

Usually every domain extension have a few available servers.

Default list is:

ac=whois.nic.ac
am=whois.nic.am
as=whois.nic.as
at=whois.aco.net
au=whois.aunic.net
be=whois.belnet.be
br=whois.registro.br
ca=whois.domainpeople.com
cc=whois.nic.cc
ch=whois.corenic.net
ck=whois.nic.ck
cl=whois.nic.cl
cn=whois.cnnic.net.cn
cr=whois.ci.ucr.ac.cr
cz=whois.cuni.cz
de=whois.fzi.de
dk=whois.uni-c.dk
ec=whois.lac.net
ee=whois.ut.ee
es=whois.eunet.es
fi=cs.hut.fi
fr=whois.nic.fr
gb=whois.easyspace.com
gb.com=whois.nomination.net
gb.net=whois.nomination.net
gs=whois.adamsnames.tc
hk=whois.cary.net
hm=whois.registry.hm
in=whois.iisc.ernet.in
is=isgate.is

it=whois.nic.it
jp=whois.discount-domain.com
kg=whois.domain.kg
kr=whois.krnic.net
kz=whois.domain.kz
li=whois.nic.li
lk=whois.nic.lk
lu=www.restena.lu
mm=whois.nic.mm
ms=whois.adamsnames.tc
mx=whois.nic.mx
nl=domain-registry.nl
no=whois.norid.no
nu=whois.nic.nu
nz=whois.domainz.net.nz
pe=whois.rcp.net.pe
pl=whois.icm.edu.pl
pt=whois.dns.pt
ru=whois.ripn.net
sc=whois.apnic.net
se=whois.nic-se.se
sg=whois.nic.net.sg
sh=whois.nic.sh
sk=whois.uakom.sk
st=whois.nic.st
tc=whois.adamsnames.tc
tf=whois.adamsnames.tc
th=whois.thnic.net
tj=whois.nic.tj
to=whois.tonic.to
tr=whois.metu.edu.tr
tw=whois.hinet.net
uk=whois.nic.uk
uk.com=whois.nomination.net
uk.net=whois.nomination.net
us=whois.alabanza.com
vg=whois.adamsnames.tc
za=whois.co.za

In internet you could find a lot of published list of available whois-servers. I suggest time-to-time to fresh a list because some whois-servers could be closed in future but newest can be opened.

For example, useful resource of available whois-servers you can find at
<ftp://sipb.mit.edu/pub/whois/whois-servers.list>



TSMInetDateTime component

[See also](#) [Methods](#)

Unit

[SMITime](#)

Description

The Time client component enables the application to request the current time and date from a remote server. The values returned may either be expressed in local time (using the local host's timezone) or system time (also known as Universal Coordinated Time or Greenwich Mean Time). The control may also be used to synchronize the local host's system clock with the remote server.

GetDateTime method

[See also Example](#)

Applies to

[TSMInetDateTime](#) component

Declaration

```
function GetDateTime: TDateTime;
```

Description

This event is a wrapper for Execute method which retrieve a required information about time from dayserver in speified format and automatically decode a value from encrypted string into TDateTime.

If you want to decrypt a string yourself, just call Execute method instead GetDateTime method.



TSMCustomWebComponent component

[See also](#) [Properties](#) [Methods](#) [Events](#)

Unit

[smiupdate](#)

Description

TSMCustomWebComponent component is an internal basic component for work using WinInet.dll

Any other components like TSMWebUpdater, TSMWebDirect, TSMRemoteImage, TSMInfoExtractor and others are inherited from this basic class.

Action property

[See also Example](#)

Applies to

[TSMCustomWebComponent](#) component

Declaration

property Action: TSMTAction;

Description

This run-time property allow to have an access to current state of connection.

You can read an Action value in any event and to check desired state (for example, when connection is occurred or remote file is found).

When current state/action is changed, automatically will be fired an OnActionChange event.

Agent property

[Example](#)

Applies to

[TSMCustomWebComponent](#) component

Declaration

```
property Agent: string;
```

Description

This is a custom description of your application which will be displayed in the log of requested web-servers.

You can place here any desired description of your application.
Default value of agent property is 'Scalabium WebUpdater'.

Typical agents of popular browsers/applications:

Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)

Mozilla/4.0 (compatible; MSIE 5.0; Windows 98)

Mozilla/4.0 (compatible; MSIE 5.0; Windows 2000) Opera 6.0 [en]

Mozilla/4.73 [en] (Win98; I)

GetRight/4.3

FlashGet

Teleport Pro/1.29.1718

Mass Downloader/2.1

OnProgress property

[See also Example](#)

Applies to

[TSMCustomWebComponent](#) component

Declaration

property OnProgress: TSMIProgress;

Description

During file downloading you can control a process in this event. You can update a progressbar or cancel a downloading, for example, ...

This event will be called after every block of retrieved information from web-server.

Password property

[See also Example](#)

Applies to

[TSMCustomWebComponent](#) component

Declaration

```
property Password: string;
```

Description

If you want to update a file from remote directory which is protected by password, you can define a password here.

Note that if you'll try to retrieve an information about file from protected directory without password, connection will be refused.

Proxy property

Applies to

[TSMCustomWebComponent](#) component

Declaration

property Proxy: [TSMInetProxy](#);

Description

Using this Proxy property you can specify any name of custom proxy server, proxy logon (user id and password), port and more. This information required for connection to remote server.

Service property

[See also Example](#)

Applies to

[TSMCustomWebComponent](#) component

Declaration

property Service: TSMIService;

Description

You must set a Service property according protocol which you want to use for file downloading.

If you want to use the HTTP-protocol, set a Service as isHTTP.

If you want to use the FTP-protocol, set a Service as isFTP.

If you want to work with shared resource in your local network, set a Service as isNetwork.

Status property

[See also Example](#)

Applies to

[TSMCustomWebComponent](#) component

Declaration

```
property Status: TSMIStatus;
```

Description

This run-time property allow to have an access to current state of file downloading.

You can read the Status value in any event and to find a current state (for example, when process is in progress or is failed by error).

When current status is changed, automatically will be fired an OnStatusChange event.

User property

[See also Example](#)

Applies to

[TSMCustomWebComponent](#) component

Declaration

```
property User: string;
```

Description

If you want to update a file from web-server which require a web authorization, you must define the UserID and password here.

Note that if you'll try to retrieve an information about file from protected directory without logon information (user and password), connection will be refused by server.

OnActionChange event

[See also Example](#)

Applies to

[TSMCustomWebComponent](#) component

Declaration

property OnActionChange: TNotifyEvent;

Description

This event is occurred automatically when Action property is changed.

In event handler you can check a desired state (for example, when connection is occurred or remote file is found).

OnAfterExecute event

See also [Example](#)

Applies to

[TSMCustomWebComponent](#) component

Declaration

property OnAfterExecute: TNotifyEvent;

Description

This event will be executed automatically in end of processing of Execute method.

OnBeforeExecute event

[See also Example](#)

Applies to

[TSMCustomWebComponent](#) component

Declaration

property OnBeforeExecute: TNotifyEvent;

Description

This event will be executed automatically when you'll call an Execute method.

OnStatusChange event

[See also Example](#)

Applies to

[TSMCustomWebComponent](#) component

Declaration

property OnStatusChange: TNotifyEvent;

Description

This event is occurred automatically when Status property is changed.

In event handler you can check a current state (for example, when process is in progress or is failed by error).



TSMCustomWebUpdater component

[See also](#) [Properties](#) [Methods](#)

Unit

[smiupdate](#)

Description

This is a basic class which implemet a work with internet connection using WinInet.dll.

This class allow to download a file by http/https/ftp protocols or local network with detailed handlers of every step.

OnFileCompare property

[See also Example](#)

Applies to

[TSMCustomWebUpdater](#) component

Declaration

property OnFileCompare: TSMIFileCompare;

Description

This event will be executed when information about remote and local files are retrieved and you may decide to skip or initialize a downloading process.

RemoteFileCreated property

[See also Example](#)

Applies to

[TSMCustomWebUpdater](#) component

Declaration

property RemoteFileCreated: TFileTime;

Description

This readonly property is readable in run-time only when information about remote file is retrieved.

You can analyze this value in OnFileCompare event, for example, when you must decide: to download a file or not.

RemoteFileName property

[See also Example](#)

Applies to

[TSMCustomWebUpdater](#) component

Declaration

```
property RemoteFileName: string;
```

Description

This value contain a name of remote file which must be retrieved from web-server for your local copy.

Note that remote filename must be relative to defined Host.

For example, if you want to download a file from

<http://www.scalabium.com/download/smdbgrid.zip> then you must define:

Service is isHTTP

Port is 80 {default port for http protocol}

Host is scalabium.com

RemoteFileName is download/smdbgrid.zip

RemoteFileSize property

[See also Example](#)

Applies to

[TSMCustomWebUpdater](#) component

Declaration

property RemoteFileSize: dWord;

Description

This readonly property is readable in run-time only when information about remote file is retrieved.

You can analyze this value in OnFileCompare event, for example, when you must decide: to download a file or not.



TSMWebUpdater component

[See also](#) [Properties](#)

Unit

[smiupdate](#)

Description

This component implements a work for automatical updating any local file by remote copy from internet by HTTP or FTP connection (using WinInet.dll) or from shared directory of your local network.



TSMWebDirect component

[See also Properties](#) [Methods](#)

Unit

[smidirect](#)

Description

The TSMWebDirect component uses your Internet connection and web browser to find and inform you of new product news, updates and patches, technical tips, and other announcements.

This is an analogue of Delphi Direct.

Format of file with news/announcements is a standard INI-file.

Sample:

```
;***** START *****  
[Scalabium News]  
http://www.scalabium.com=01/14/2002 - New product (SMInternet suite for  
Delphi/C++Builder) is released  
http://www.scalabium.com/faq/dc_tips.htm=12/20/2001 - A few new tips added  
  
[Delphi tips]  
http://www.scalabium.com/faq/dc_tips.htm=List of tips by category  
  
[SMExport suite]  
http://www.scalabium.com/sme/index.htm1=01/06/2001 - Added the PDF format exporting  
http://www.scalabium.com/sme/index.htm2=12/20/2001 - Added the SPSS file format for  
statistical software support  
http://www.scalabium.com/sme/index.htm=SMExport suite page  
http://www.scalabium.com/download/sme/smetrial.zip=SMExport trial packages  
http://www.scalabium.com/download/sme/smedemo.zip=SMExport demo examples  
http://www.scalabium.com/smorder.htm=SMExport registration  
;***** END *****
```

PS: the sample file you can download at <http://www.scalabium.com/scalabium.sin>

ShowMode property

[See also Example](#)

Applies to

[TSMWebDirect](#) component

Declaration

property ShowMode: TSMIShowMode;

Description

ShowMode property specify when dialog with downloaded news will be displayed.

Title property

See also [Example](#)

Applies to

[TSMWebDirect](#) component

Declaration

```
property Title: string;
```

Description

You can define any text as caption of dialog which will be displayed with a news tree.

Show method

[See also Example](#)

Applies to

[TSMWebDirect](#) component

Declaration

```
procedure Show;
```

Description

You can show a dialog with a news tree without downloading of current file from remote server. In this case will be displayed news which is stored in local file with records which was downloaded in previous times...

Dialog will be displayed as non-modal window.

ShowModal method

[See also Example](#)

Applies to

[TSMWebDirect](#) component

Declaration

```
procedure ShowModal;
```

Description

You can show a dialog with a news tree without downloading of current file from remote server. In this case will be displayed news which is stored in local file with records which was downloaded in previous times...

Dialog will be displayed as modal window.



TSMUPSTracker component

[See also](#) [Properties](#) [Methods](#)

Unit

[SMIUPS](#)

Description

This component allows to track your package via UPS-Online service. You can request a last state of your package or to retrieve a full list with detailed steps for your package.

TrackingNumber property

[See also Example](#)

Applies to

[TSMUPSTracker](#) component

Declaration

```
property TrackingNumber: string;
```

Description

This property contain a value of package number which you want to trace via online service of UPS.

Just provide a number in format which you retrieved from UPS deparment. For example, w607 8836 707.

TrackingType property

[See also](#)

Applies to

[TSMUPSTracker](#) component

Declaration

property TrackingType: TSMUPSTrackingType;

Description

This property contain a type of desired package tracking. You can retrieve a short information of current state or detailed information about every step in package deployment.

By default, you'll retrieve a short information for last step.

GetActivityAsDateTime method

[See also](#)

Applies to

[TSMUPSTracker](#) component

Declaration

```
function GetActivityAsDateTime: TDateTime;
```

Description

This method allow to convert a datetime value from remote UPS format into TDateTime type.

GetUPSStatus method

[See also](#)

Applies to

[TSMUPSTracker](#) component

Declaration

```
function GetUPSStatus(i: TSMUPSStatus): string;
```

Description

This method allow to extract from retrieved information about package a required information.

For example, you can extract an information about current location of package or weight of package.



TSMInfoExtractor component

[See also](#) [Properties](#) [Methods](#)

Unit

[smixtract](#)

Description

This component allow to extract any information from online resources.

For example, you can extract a list of emails which are published in web-page or receive a list of pictures (or archives) which are displayed (linked) in this page.

You can customize a lot of parameters for data extraction - prefix, wildcard and more

Extract property

[See also Example](#)

Applies to

[TSMInfoExtractor](#) component

Declaration

property Extract: TSMIExtract;

Description

Using this property you can define what kind of data do you want to extract from online resource.

ExtractedInfo property

[See also Example](#)

Applies to

[TSMInfoExtractor](#) component

Declaration

```
property ExtractedInfo: TStrings;
```

Description

When data is extracted from online resource, parsed data is available from this property. You can iterate thru list of parsed values which are produced by your custom criteria.

ParseData method

Applies to

[TSMInfoExtractor](#) component

Declaration

```
procedure ParseData;
```

Description

This is an internal method which allow to parse downloaded resource and extract from this data any requested information.



TSMRemoteImage component

[See also](#) [Properties](#) [Methods](#)

Unit

[smiimage](#)

Description

TSMRemoteImage component allow to display of remote image and automatically update an image by some interval.

This is very useful when you want to display an image from remote web-camera or to view dynamic resources (log tracker, stock information), for example.

Active property

[See also Example](#)

Applies to

[TSMRemoteImage](#) component

Declaration

property Active: Boolean;

Description

Active property controls whether the remote image refresh every defined interval.

Use Active to enable or disable the image refresh. If Active is True, the remote image will be refreshed after interval is occurred.

If Active is False, the image will be not refreshed automatically.

The default is True.

Image property

[See also Example](#)

Applies to

[TSMRemoteImage](#) component

Declaration

property Image: TImage;

Description

Using this property you can specify any existing TImage component which will be used for image displaying of downloaded remote image.

Interval property

See also [Example](#)

Applies to

[TSMRemoteImage](#) component

Declaration

property Interval: Integer;

Description

Interval determines the amount of time, in milliseconds, that passes before the remote image will be refreshed (downloaded a new copy).

Interval determines how frequently the image downloading occurs. Each time the specified interval passes, the new copy of remote image is downloaded.

Start method

[See also](#)

Applies to

[TSMRemoteImage](#) component

Declaration

```
procedure Start;
```

Description

Activate an image downloading.

Note

This is a same as Active := True

Stop method

[See also](#)

Applies to

[TSMRemoteImage](#) component

Declaration

```
procedure Stop;
```

Description

Deactivate an image downloading.

Note

This is a same as Active := False



TSMIHTTPClient component

[See also](#) [Properties](#) [Methods](#)

Unit

[SMIHTTP](#)

Description

TSMIHTTPClient component allow to work with any http-server using HTTP-protocol (Hypertext Tranfer Protocol).

HTTP v 1.0 and v1.1 are supported.

You can implement any http client features - send request, post data and more. For example, you can implement web browser.

Action property

[See also Example](#)

Applies to

[TSMIHTTPClient](#) component

Declaration

property Action: TSMIHTTPAction;

Description

Define a type of request which will be sent to server.

Headers property

[See also Example](#)

Applies to

[TSMIHTTPClient](#) component

Declaration

property Headers: TStrings;

Description

Allow to define any custom values in header of request which will be sent to remote server.

For example, you can define there parameters for your CGI-script.

RemoveHeaders property

[See also](#)

Applies to

[TSMIHTTPClient](#) component

Declaration

property RemoveHeaders: Boolean;

Description

This property allow to remove any headers which are produced by remote server from retrieved data.

If RemoveHeaders is True, then Execute method will return pure data without system headers.

Else if RemoveHeaders is False, then Execute method will return any data which are received from remote server including server-side headers.

Note that in any case you can read received headers in ResponseHeaders property.

Request property

[See also Example](#)

Applies to

[TSMIHTTPClient](#) component

Declaration

property Request: **string**;

Description

This property is a name of your CGI-script which you would like to call in server. There you must define a full path to script excluding domain name.

ResponseHeaders property

[See also Example](#)

Applies to

[TSMIHTTPClient](#) component

Declaration

property ResponseHeaders: TStrings;

Description

This is a list of detailed response headers which are received from remote-server. Usually it contain some system information like data encoding, date of last file changing, length of asked data, return code etc

Generally this list is controled by server and you can use it in readonly mode only.

GetResponse method

[Example](#)

Applies to

[TSMIHTTPClient](#) component

Declaration

```
function GetResponse(Value: string): string;
```

Description

This method allow to retrieve required value from list of headers, which are produced by remote server.

For example, you can ask server encoding or date of last changing for required file.



TSMRemoteFiles component

[See also](#) [Properties](#) [Methods](#)

Unit

[smidir](#)

Description

TSMRemoteFiles component which allow to load a list of files from remote web-server or shared folder in local network. You can specify desired directory and file wildcard. Also you can define either load data from child sub-folders or load only from specified folder.

Directory property

[See also Example](#)

Applies to

[TSMRemoteFiles](#) component

Declaration

```
property Directory: string;
```

Description

Specify a name of remote directory which will be processed.

If SubFolders is True, then file names will be retrieved from this directory and every sub-directory.

Else if SubFolders is False, then file names will be retrieved from this directory only.

FileList property

See also [Example](#)

Applies to

[TSMRemoteFiles](#) component

Declaration

```
property FileList: TStrings;
```

Description

This run-time property allow to read a result of file processing. Just read a list of files which are exist in remote directory.

SubFolders property

[See also Example](#)

Applies to

[TSMRemoteFiles](#) component

Declaration

property SubFolders: Boolean;

Description

If SubFolders is True, then file names will be retrieved from this directory and every sub-directory.

Else if SubFolders is False, then file names will be retrieved from this directory only.

Wildcard property

[See also Example](#)

Applies to

[TSMRemoteFiles](#) component

Declaration

```
property WildCard: string;
```

Description

Specify a wildcard for file names which you want to read from remote directory.

For example, to read archives, set WildCard := '*.zip'.

You can use any general characters for mask which are accessible for wildcards.

TSMInetProxy class

[Properties](#) [Methods](#)

Unit

[SMInBase](#)

Description

This class declare the complex property for proxy support.
This information required for connection to remote server.

Password property

[See also Example](#)

Applies to

[TSMInetProxy](#) object

Declaration

```
property Password: string;
```

Description

Specify the password that will be used as logon to proxy server.

Port property

See also [Example](#)

Applies to

[TSMInetProxy](#) object

Declaration

```
property Port: Integer;
```

Description

Specify the proxy port that will be used as logon to proxy server.

Server property

[See also Example](#)

Applies to

[TSMInetProxy](#) object

Declaration

```
property Server: string;
```

Description

Specify the proxy server name that will be used as logon to internet.

UseProxy property

[See also Example](#)

Applies to

[TSMInetProxy](#) object

Declaration

property UseProxy: Boolean;

Description

Specify either component will use proxy settings for connection.

If UseServer is True, then Server value will be applied.

Else if UseServer is False, then Server value will be not used at all.

User property

See also [Example](#)

Applies to

[TSMInetProxy](#) object

Declaration

```
property User: string;
```

Description

Specify the user id that will be used as logon to proxy server.

FillProxyInfo method

Applies to

[TSMInetProxy](#) object

Declaration

```
procedure FillProxyInfo;
```

Description

This procedure allow to read the default proxy settings from MS Windows.

TSMFedExTracker component

[Properties](#) [Methods](#)

Unit

[SMIFedEx](#)

Description

This component allows to track your package via FedEx online service. You can request a last state of your package or to retrieve a full list with detailed activities for your package.

Activity property

[See also Example](#)

Applies to

[TSMFedExTracker](#) component

Declaration

property Activity: TStrings;

Description

This run-time property allow to read a list of all activities with details for specified package

ShipmentInfo property

[See also Example](#)

Applies to

[TSMFedExTracker](#) component

Declaration

```
property ShipmentInfo: TStrings;
```

Description

This run-time property allow to read a list of all available properties about shipment (service type, ship and delivery date-time, reference, signed by etc)

TrackingNumber property

[See also Example](#)

Applies to

[TSMFedExTracker](#) component

Declaration

```
property TrackingNumber: string;
```

Description

This property contain a value of package number which you want to trace via online service of FedEx.

Just provide a number in format which you retrieved from FedEx depart. For example, 040862810001923.

TSMWeatherInfo component

[Properties](#) [Methods](#)

Unit

[SMIWeath](#)

Description

This component allows to get an information about wheather from online resources by ZIP/Postal Code.

Temperature property

[Example](#)

Applies to

[TSMWeatherInfo](#) component

Declaration

property Temperature: Integer;

Description

This is read-only property with result of online tracking. Read this value for retrieved temperature by your request.

TrackingType property

[Example](#)

Applies to

[TSMWeatherInfo](#) component

Declaration

property TrackingType: TSMWeatherTrackingType;

Description

This property contain a type of desired temperature tracking.

PS: current version supports the tracking by ZIP only

Zip property

[Example](#)

Applies to

[TSMWeatherInfo](#) component

Declaration

```
property Zip: string;
```

Description

Apply any ZIP code as value for this property which you want to trace via online service and get a temperature.

TSMInetStatusDialog type

[Properties](#) [Methods](#)

Unit

[smiupdate](#)

Description

This type allow to define any custom settings for standard visual status dialog that could be displayed during file update/downloading.

You may customize any label or picture in this dialog.

Description property

[Example](#)

Applies to

[TSMInetStatusDialog](#) object

Declaration

```
property Description: string;
```

Description

Description property specifies the caption text that appears for status dialog as long text with note to end-users.

Set the Description property to provide the custom note. If Description is not set (empty), default text will be used.

Enabled property

[Example](#)

Applies to

[TSMInetStatusDialog](#) object

Declaration

property Enabled: Boolean;

Description

If Enabled property is True then standard status dialog will be displayed.

Else if Enabled property is False, this dialog will be not shown. You may set the Enabled = False if you use own status dialog, for example.

HelpContext property

Applies to

[TSMInetStatusDialog](#) object

Declaration

```
property HelpContext: THelpContext;
```

Description

HelpContext is a context ID for the entire status dialog.
Use this property to associate a help screen.

Icon property

[Example](#)

Applies to

[TSMInetStatusDialog](#) object

Declaration

```
property Icon: TIcon;
```

Description

Icon property specifies the icon that appears for status dialog (in caption of form).

Set the Icon property to provide an icon for the dialog. If Icon is not set, default icon will be used.

Picture property

[Example](#)

Applies to

[TSMInetStatusDialog](#) object

Declaration

property Picture: TPicture;

Description

Picture property specifies the logo that appears for status dialog.

Set the Picture property to provide the custom image. If Picture is not set, default logo will be used.

Title property

[Example](#)

Applies to

[TSMInetStatusDialog](#) object

Declaration

```
property Title: TCaption;
```

Description

Title property specifies the caption text that appears for status dialog.

Set the Title property to provide the custom text. If Text is not set (empty), default text will be used.

See also

[TSMWhols](#)

[TSMInetDateTime](#)

[TSMUPSTracker](#)

Properties

► Run-time only ■ Key properties

- [About](#)
- [Address](#)
- ■ [ClientType](#)
- [Host](#)
- [KeepConnection](#)
- [Port](#)
- [Proxy](#)

Methods

- Key methods

Create{linkDelphi=Create_Method}

Destroy{linkDelphi=Destroy_Method}

- [Execute](#)

- [GetQuery](#)

- [Connect](#)

- [Disconnect](#)

Events

- Key events
- [OnAfterExecute](#)
- [OnBeforeExecute](#)
- [OnConnect](#)
- [OnConnecting](#)
- [OnDisconnect](#)
- [OnError](#)
- [OnLookup](#)
- [OnReceiveTextFromSocket](#)
- [OnSendTextToSocket](#)

SMinBase unit

In this unit you will find a description of basic socket class. Any other components of suite (like TSMWhols, TSMInetDateTime and TSMUPSTracker) are inherited from this class.

Components

[TSMInetBase](#)

KeepConnection property example

```
SMWhols.KeepConnection := True;
```

```
{process a first request}  
SMWhols.Domain := 'microsoft.com';  
memo.Lines.Add(SMWhols.Execute);
```

```
{process a second request}  
SMWhols.Domain := 'borland.com';  
memo.Lines.Add(SMWhols.Execute);
```

```
{process a third request}  
SMWhols.Domain := 'scalabium.com';  
memo.Lines.Add(SMWhols.Execute);
```

```
{disconnect from Internet}  
SMWhols.Disconnect;
```

See also

[Connect](#)
[Disconnect](#)
[GetQuery](#)

Execute method example

```
{process a domain checking}  
SMWhoIs.Domain := 'scalabium.com';
```

```
{retrieve a result}  
s := SMWhoIs.Execute;
```

```
{disconnect from Internet}  
SMWhoIs.Disconnect;
```

```
{show a result of check}  
ShowMessage(s);
```

See also

[Execute](#)

GetQuery method example

```
function TSMWhols.GetQuery: string;  
const  
  CRLF = #13#10;  
begin  
  Result := Domain + CRLF  
end;
```

See also

[OnConnecting](#)

[OnConnect](#)

See also

[OnDisconnect](#)

See also

[Execute](#)

[OnBeforeExecute](#)

See also

[Execute](#)
[OnAfterExecute](#)

OnBeforeExecute event example

```
begin  
  {show a text in status bar}  
  StatusBar.Text := 'Please wait...';  
end;
```

See also

[OnConnecting](#)
[Connect](#)

OnConnect event example

```
begin  
  {activate Cancel button}  
  btnCancel.Enabled := True;  
end;
```

See also

[Connect](#)
[OnConnect](#)

OnConnecting event example

```
begin  
  {update status bar text}  
  StatusBar.Panels[0].Text := 'Connecting...';  
end;
```

See also

[OnConnecting](#)

[OnConnect](#)

[Disconnect](#)

OnDisconnect event example

```
begin  
  {disable a Cancel button}  
  btnCancel.Enabled := False;  
end;
```

OnError event example

```
procedure TfrmIOLink.SMInetError(Sender: TObject;  
Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;  
var ErrorCode: Integer);  
var  
    strError: string;  
begin  
    case ErrorEvent of  
        eeGeneral: strError := 'The socket received an error message that does not fit into any of  
the following categories.';  
        eeSend: strError := 'An error occurred when trying to write to the socket connection.';  
        eeReceive: strError := 'An error occurred when trying to read from the socket connection.';  
        eeConnect: strError := 'A connection request that was already accepted could not be  
completed.';  
        eeDisconnect: strError := 'An error occurred when trying to close a connection.';  
        eeAccept: strError := 'A problem occurred when trying to accept a client connection  
request.';  
    end;  
    lblError.Caption := IntToStr(ErrorCode) + ' ' + strError;  
    ErrorCode := 0;  
end;
```

See also

[TSMInetBase](#)

[TSMInetDateTime](#)

[TSMUPSTracker](#)

Properties

- ▶ Run-time only
- Key properties
 - [AutoSelectServer](#)
 - [Domain](#)
 - [ServerList](#)

Methods

- Key methods

Create{linkDelphi=Create_Method}

Destroy{linkDelphi=Destroy_Method}

GetQuery{linkDelphi=GetQuery_Method}

SMWhols unit

In this unit you will find a declaration of TSMWhols component.

Components

[TSMWhols](#)

Constants

DefaultWholsServer

See also

[Domain](#)
[ServerList](#)
[Host](#)

AutoSelectServer property example

```
SMWhoIs.AutoSelectServer := False;  
SMWhoIs.Host := 'whois.internic.net';  
SMWhoIs.Domain := 'scalabium.com';  
str1 := SMWhoIs.Execute;
```

```
SMWhoIs.AutoSelectServer := True;  
SMWhoIs.Domain := 'scalabium.com';  
str2 := SMWhoIs.Execute;
```

```
ShowMessage(str1 + #13#10 + str2);
```


See also

[AutoSelectServer](#)
[ServerList](#)
[Host](#)

Domain property example

```
SMWhols.AutoSelectServer := True;  
SMWhols.Domain := 'scalabium.com';  
ShowMessage(SMWhols.Execute);
```

See also

[AutoSelectServer](#)

[Domain](#)

[Host](#)

See also

[TSMInetBase](#)

[TSMWhols](#)

[TSMUPSTracker](#)

Methods

- Key methods

Create{linkDelphi>Create_Method}

- [GetDateTime](#)

SMITime unit

In this unit you will find a declaration of TSMInetDateTime component.

Components

[TSMInetDateTime](#)

Constants

DefaultDTServer

See also

[Execute](#)

GetDateTime method example

```
SMInetDateTime.Port := IPPORT_DAYTIME;  
SMInetDateTime.Host := 'www.boulder.nist.gov';
```

```
ShowMessage(FormatDateTime('dddd, mmmm dd, yyyy, tt ',  
SMInetDateTime.GetDateTime()));
```


See also

[TSMCustomWebUpdater](#)

[TSMWebUpdater](#)

[TSMWebDirect](#)

[TSMInfoExtractor](#)

[TSMRemoteImage](#)

[TSMIHTTPClient](#)

[TSMRemoteFiles](#)

Properties

- ▶ Run-time only ■ Key properties
- ▶ ■ [Action](#)
- [Agent](#)
- [Host](#)
- [OnProgress](#)
- [Password](#)
- [Port](#)
- [Proxy](#)
- [Service](#)
- ▶ ■ [Status](#)
- [User](#)
- [StatusDialog](#)

Methods

- Key methods

Create{linkDelphi>Create_Method}

- [Execute](#)

Events

- Key events
- [OnActionChange](#)
- [OnAfterExecute](#)
- [OnBeforeExecute](#)
- [OnStatusChange](#)

smiupdate unit

In this unit you will find a declaration of TSMCustomWebComponent type which is a parent class for any component which uses a WinInet.dll.

Components

[TSMCustomWebComponent](#)

[TSMCustomWebUpdater](#)

[TSMWebUpdater](#)

Types

[TSMIAction](#)

[TSMIFileCompare](#)

[TSMIProgress](#)

[TSMIService](#)

[TSMIStatus](#)

[TSMInetStatusDialog](#)

See also

[OnActionChange](#)
[TSMIAction](#)

Action property example

```
if (SMWebDirect.Action = iaFileFound) then  
  ShowMessage('Remote file is found.')
```

Agent property example

SMWebUpdater.Agent := Application.Title;

or

SMWebUpdater.Agent := 'Advanced Web Checker';

See also

[TSMIProgress](#)

[Execute](#)

[OnBeforeExecute](#)

[OnAfterExecute](#)

[OnStatusChange](#)

[OnActionChange](#)

OnProgress property example

```
{refresh a progressbar}  
pbStatus.Max := MaxValue;  
pbStatus.Position := CurValue;
```

```
{update a caption with current value of retrieved bytes}  
lblProgress.Caption := 'Downloaded ' + IntToStr(CurValue) + ' of ' + IntToStr(MaxValue) + '  
bytes (' + IntToStr(CurValue*100 div MaxValue) + '%)';
```

```
{if Cancel button is pressed, then Abort}  
Abort := IsCanceled();
```

```
{allow to repaint a window}  
Application.ProcessMessages;
```

See also

[User](#)

Password property example

```
SMWebUpdater.User := 'peter';  
SMWebUpdater.Password := 'p1gRsTYS0k';
```

See also

[TSMIService](#)

[Host](#)

[Port](#)

Service property example

```
SMWebUpdater.Service := isHTTP;  
SMWebUpdater.Host := 'scalabium.com';
```

See also

[OnStatusChange](#)
[TSMIStatus](#)

Status property example

```
if (SMWebDirect.Status = isFailed) then  
  ShowMessage('Update is failed!')
```


See also

[Password](#)

User property example

```
SMWebUpdater.User := 'peter';  
SMWebUpdater.Password := 'p1gRsTYS0k';
```

See also

[Execute](#)

[OnBeforeExecute](#)

[OnAfterExecute](#)

[OnStatusChange](#)

OnActionChange event example

```
procedure TfrmMain.SMWebDirectActionChange(Sender: TObject);
const
arrAction: array[TSMIAction] of string = (
'Idle',
'Connecting...',
'Connected.',
'Disconnecting...',
'Disconnected.',
'File search...',
'File found.',
'Comparing...',
'Compared.',
'Downloading...',
'Downloaded.');
```

begin
lblProgress.Caption := arrAction[SMWebDirect.Action];

Application.ProcessMessages
end;

See also

[Execute](#)
[OnBeforeExecute](#)

[OnActionChange](#)
[OnStatusChange](#)

OnAfterExecute event example

```
case SMWebDirect.Status of
isFinished: s := 'File updated succesfully';
isFailed: s := 'Error is occured during file updating.';
isSkipped: s := 'File is not changed';
else
s := "";
end;

if (s <> "") then
ShowMessage(s)
```

See also

[Execute](#)

[OnAfterExecute](#)

[OnActionChange](#)

[OnStatusChange](#)

OnBeforeExecute event example

```
WriteToLog(Now(), 'update starting...');
```


See also

[Execute](#)

[OnBeforeExecute](#)

[OnAfterExecute](#)

[OnActionChange](#)

OnStatusChange event example

```
procedure TfrmMain.SMWebDirectStatusChange(Sender: TObject);
const
  arrStatus: array[TSMIStatus] of string = (
    'In progress...',
    'Completed.',
    'Failed.',
    'Skipped',
    'File not found');
begin
  lblProgress.Caption := arrStatus[SMWebDirect.Status];

  Application.ProcessMessages
end;
```

See also

[TSMWebUpdater](#)

[TSMWebDirect](#)

[TSMInfoExtractor](#)

[TSMRemoteImage](#)

[TSMHTTPClient](#)

[TSMRemoteFiles](#)

Properties

- ▶ Run-time only ■ Key properties
- [OnFileCompare](#)
- ▶ ■ [RemoteFileCreated](#)
- [RemoteFileName](#)
- ▶ ■ [RemoteFileSize](#)
- [Proxy](#)

Methods

- Key methods

`InternalExecute(linkDelphi=InternalExecute_Method)`

See also

[RemoteFileName](#)

[RemoteFileCreated](#)

[RemoteFileSize](#)

[TSMIFileCompare](#)

OnFileCompare property example

```
procedure TForm1.SMWebUpdater1FileCompare(Sender: TObject;  
var Equal: Boolean);  
var  
dt: TDateTime;  
st: TSystemTime;  
begin  
  {to convert a TFileTime into TSystemTime}  
  FileTimeToSystemTime(SMWebUpdater1.RemoteFileCreated, st);  
  
  {to convert a TSystemTime into TDateTime}  
  dt := EncodeDate(st.wYear, st.wMonth, st.wDay) +  
    EncodeTime(st.wHour, st.wMinute, st.wSecond, st.wMilliseconds);  
  
  {if remote file is created before July 1, 2002 then don't update a local file}  
  if (dt < EncodeDate(2002, 07, 01)) then  
    Equal := True  
end;
```

See also

[RemoteFileSize](#)
[RemoteFileName](#)

[OnFileCompare](#)

RemoteFileCreated property example

```
var
dt: TDateTime;
st: TSystemTime;
begin
{to convert a TFileTime into TSystemTime}
FileTimeToSystemTime(SMWebUpdater1.RemoteFileCreated, st);

{to convert a TSystemTime into TDateTime}
dt := EncodeDate(st.wYear, st.wMonth, st.wDay) +
EncodeTime(st.wHour, st.wMinute, st.wSecond, st.wMilliseconds);
end;
```

See also

[RemoteFileSize](#)

[RemoteFileCreated](#)

[OnFileCompare](#)

RemoteFileName property example

SMWebUpdater.RemoteFileName := 'download/smdbgrid.zip';

SMWebUpdater.LocalFileName := 'C:\ARCHIVE\DELPHI\Scalabium\smdbgrid.zip';

See also

[RemoteFileName](#)

[RemoteFileCreated](#)

[OnFileCompare](#)

RemoteFileSize property example

```
ShowMessage('Size of remote file is ' + IntToStr(SMWebUpdater.RemoteFileSize) + '
byte(s)')
```

See also

[TSMCustomWebUpdater](#)

[TSMWebDirect](#)

[TSMInfoExtractor](#)

[TSMRemoteImage](#)

[TSMHTTPClient](#)

[TSMRemoteFiles](#)

Properties

- ▶ Run-time only
- Key properties

`LocalFileName` {linkDelphi=LocalFileName_Property}

See also

[TSMWebUpdater](#)
[TSMInfoExtractor](#)
[TSMRemoteImage](#)
[TSMHTTPClient](#)
[TSMRemoteFiles](#)

Properties

- ▶ Run-time only
- Key properties
 - [ShowMode](#)
 - [Title](#)

Methods

- Key methods

Create{linkDelphi=Create_Method}

Destroy{linkDelphi=Destroy_Method}

Execute{linkDelphi=Execute_Method}

- [Show](#)

- [ShowModal](#)

smidirect unit

In this unit you will find a declaration of TSMWebDirect component.

Components

[TSMWebDirect](#)

Types

[TSMIShowMode](#)

See also

[TSMIShowMode](#)
[Show](#)
[ShowModal](#)

ShowMode property example

```
SMWebDirect.ShowMode := omIfChnaged;  
SMWebDirect.Execute
```

See also

[Execute](#)

[Show](#)

[ShowModal](#)

[ShowMode](#)

Title property example

SMWebDirect.Title := 'Hot News from World of Delphi Components';

See also

[Title](#)

[Execute](#)

[ShowModal](#)

Show method example

```
SMWebDirect.Show;
```

See also

[Title](#)
[Execute](#)
[Show](#)

ShowModal method example

```
SMWebDirect.ShowModal;
```

See also

[TSMInetBase](#)

[TSMWhols](#)

[TSMInetDateTime](#)

Properties

- ▶ Run-time only
- Key properties
 - [TrackingNumber](#)
 - [TrackingType](#)

Methods

- Key methods

Create{linkDelphi=Create_Method}

- [GetActivityAsDateTime](#)

GetQuery{linkDelphi=GetQuery_Method}

- [GetUPSStatus](#)

AfterExecute{linkDelphi=AfterExecute_Method}

SMIUPS unit

In this unit you will find a declaration of TSMUPSTracker component.

Components

[TSMUPSTracker](#)

Types

[TSMUPSStatus](#)

[TSMUPSTrackingType](#)

See also

[TrackingType](#)

[GetActivityAsDateTime](#)

[GetUPSStatus](#)

TrackingNumber property example

```
{define a package number}
SMUPSTracker.TrackingNumber := 'w607 8836 707';

{type of tracking}
SMUPSTracker.TrackingType := upsCurrentStatus;

{track a package}
SMUPSTracker.Execute;

{show results}
MemoResults.Lines.Text := "";
MemoResults.Lines.Add('Tracking Number: ' +
SMUPSTracker.GetUPSSStatus(ssTrackingNumber));
MemoResults.Lines.Add('Last Activity: ' + FormatDateTime('dddd, mmm dd, yyyy, tt',
SMUPSTracker.GetActivityAsDateTime));
MemoResults.Lines.Add('Last Status: ' + SMUPSTracker.GetUPSSStatus(ssLastStatus));
MemoResults.Lines.Add("");
MemoResults.Lines.Add('Service Type: ' + SMUPSTracker.GetUPSSStatus(ssServiceType));
MemoResults.Lines.Add('Weight: ' + SMUPSTracker.GetUPSSStatus(ssWeight) + ' ' +
SMUPSTracker.GetUPSSStatus(ssWeightUnits));
MemoResults.Lines.Add('Signed By: ' + SMUPSTracker.GetUPSSStatus(ssSignedBy));
MemoResults.Lines.Add('Location: ' + SMUPSTracker.GetUPSSStatus(ssLocation));
```

See also

[TrackingNumber](#)

[GetActivityAsDateTime](#)

[GetUPSStatus](#)

[TSMTrackingType](#)

See also

[TrackingNumber](#)

[TrackingType](#)

[GetUPSStatus](#)

See also

[TrackingNumber](#)

[TrackingType](#)

[GetActivityAsDateTime](#)

[TSMUPSStatus](#)

See also

[TSMWebUpdater](#)

[TSMWebDirect](#)

[TSMRemoteImage](#)

[TSMHTTPClient](#)

[TSMRemoteFiles](#)

Properties

- ▶ Run-time only ■ Key properties
- [Extract](#)
- ▶ ■ [ExtractedInfo](#)

Methods

- Key methods

Create{linkDelphi=Create_Method}

Destroy{linkDelphi=Destroy_Method}

Execute{linkDelphi=Execute_Method}

- [ParseData](#)

smixtract unit

In this unit you will find a declaration of TSMInfoExtractor component for data extraction from online resources.

Components

[TSMInfoExtractor](#)

Types

[TSMExtract](#)

See also

[TSMIExtract](#)
[ExtractedInfo](#)

Extract property example

```
yourSMInfoExtractor.Extract := ieMailTo;  
if yourSMInfoExtractor.Execute then  
MemoResults.Lines.Assign(yourSMInfoExtractor.ExtractedInfo)
```

See also

[Extract](#)

[TSMExtract](#)

ExtractedInfo property example

```
yourSMInfoExtractor.Extract := ieMailTo;  
if yourSMInfoExtractor.Execute then  
MemoResults.Lines.Assign(yourSMInfoExtractor.ExtractedInfo)
```

See also

[TSMWebUpdater](#)

[TSMWebDirect](#)

[TSMInfoExtractor](#)

[TSMHTTPClient](#)

[TSMRemoteFiles](#)

Properties

- ▶ Run-time only
- Key properties
 - [Active](#)
 - [Image](#)
 - [Interval](#)

Methods

- Key methods

Create{linkDelphi=Create_Method}

Destroy{linkDelphi=Destroy_Method}

Execute{linkDelphi=Execute_Method}

- [Start](#)

- [Stop](#)

smiimage unit

In this unit you will find a declaration of TSMRemoteImage component for displaying of remote image. Very useful when you want to display an image from remote web-camera, for example.

Components

[TSMRemoteImage](#)

See also

[Start](#)

[Stop](#)

Active property example

```
yourSMRemoteImage.RemoteFileName := edRemoteFileName.Text;  
yourSMRemoteImage.Interval := 30000;  
yourSMRemoteImage.Active := True;
```

See also

[Active Interval](#)

Image property example

```
yourSMRemoteImage.Image := Form1.Image1;
```

See also

[Active](#)

[Image](#)

Interval property example

```
{to refresh an image every 30 seconds}  
yourSMRemoteImage.Interval := 30000;
```

See also

[Active](#)

[Stop](#)

See also

[Active](#)

[Stop](#)

See also

[TSMWebUpdater](#)

[TSMWebDirect](#)

[TSMInfoExtractor](#)

[TSMRemoteImage](#)

[TSMRemoteFiles](#)

Properties

► Run-time only ■ Key properties

■ [Action](#)

■ [Headers](#)

~~ProxyPassword~~{linkDelphi=ProxyPassword_Property}

~~ProxyServer~~{linkDelphi=ProxyServer_Property}

~~ProxyUser~~{linkDelphi=ProxyUser_Property}

■ [RemoveHeaders](#)

■ [Request](#)

► ■ [ResponseHeaders](#)

Methods

- Key methods

Create{linkDelphi=Create_Method}

Destroy{linkDelphi=Destroy_Method}

GetQuery{linkDelphi=GetQuery_Method}

- [GetResponse](#)

AfterExecute{linkDelphi=AfterExecute_Method}

SMIHTTP unit

In this unit you will find a declaration of TSMIHTTPClient component with implementation of simple http-client.

Components

[TSMIHTTPClient](#)

Types

[TSMIHTTPAction](#)

See also

[TSMIHTTPAction](#)
[Request](#)
[RemoveHeaders](#)
[ResponseHeaders](#)

Action property example

{sample for POST action with parameters for CGI-script}

```
SMIHTTPClient.Headers.Add('Accept: text/html, */*');
SMIHTTPClient.Headers.Add('User-Agent: Scalabium HTTPClient');
SMIHTTPClient.Headers.Add("");
strParams :=
'recipient=mshkolnik@yahoo.com&Module=SMInternet&Version=1.40&subject=BugReport';
SMIHTTPClient.Headers.Add(strParams);
SMIHTTPClient.Headers.Insert(0, 'Content-Length: ' + IntToStr(Length(strParams)));
SMIHTTPClient.Action := haPost;
```

{show an answer from http-server}

```
MemoHTTP.Lines.Text := SMIHTTPClient.Execute;
```

See also

[RemoveHeaders](#)
[ResponseHeaders](#)

Headers property example

{sample for POST action with parameters for CGI-script}

```
SMIHTTPClient.Headers.Add('Accept: text/html, */*');
SMIHTTPClient.Headers.Add('User-Agent: Scalabium HTTPClient');
SMIHTTPClient.Headers.Add("");
strParams :=
'recipient=mshkolnik@yahoo.com&Module=SMInternet&Version=1.40&subject=BugReport';
SMIHTTPClient.Headers.Add(strParams);
SMIHTTPClient.Headers.Insert(0, 'Content-Length: ' + IntToStr(Length(strParams)));
SMIHTTPClient.Action := haPost;
```

{show an answer from http-server}

```
MemoHTTP.Lines.Text := SMIHTTPClient.Execute;
```


See also

[Request](#)

[Headers](#)

[ResponseHeaders](#)

See also

[RemoveHeaders](#)
[Headers](#)
[ResponseHeaders](#)

Request property example

```
SMIHTTPClient.Host := 'scalabium.com';  
SMIHTTPClient.Request := '/cgi-bin/validate.pl';  
SMIHTTPClient.Action := haPost;
```

```
SMIHTTPClient.Execute;
```

See also

[Request](#)

[Headers](#)

[RemoveHeaders](#)

[GetResponse](#)

ResponseHeaders property example

{display detailed response header which is received from http-server}

```
for i := 0 to SMIHTTPClient.ResponseHeaders.Count-1 do
begin
s := SMIHTTPClient.ResponseHeaders[i];
j := Pos(':', s);
if j > 0 then
begin
sgResponse.Cells[0, i+1] := Copy(s, 1, j-1);
sgResponse.Cells[1, i+1] := Copy(s, j+1, Length(s));
end
else
sgResponse.Cells[1, i+1] := s;
end;
```

GetResponse method example

{display detailed response header which is received from http-server}

```
for i := 0 to SMIHTTPClient.ResponseHeaders.Count-1 do
begin
s := SMIHTTPClient.ResponseHeaders[i];
j := Pos(':', s);
if j > 0 then
begin
sgResponse.Cells[0, i+1] := Copy(s, 1, j-1);
sgResponse.Cells[1, i+1] := Copy(s, j+1, Length(s));
end
else
sgResponse.Cells[1, i+1] := s;
end;
```

See also

[TSMWebUpdater](#)

[TSMWebDirect](#)

[TSMInfoExtractor](#)

[TSMRemoteImage](#)

[TSMHTTPClient](#)

Properties

- ▶ Run-time only ■ Key properties
 - [Directory](#)
 - ▶ ■ [FileList](#)
 - [SubFolders](#)
 - [WildCard](#)

Methods

- Key methods

Create{linkDelphi=Create_Method}

Destroy{linkDelphi=Destroy_Method}

InternalExecute{linkDelphi=InternalExecute_Method}

smidir unit

In this unit you will find a declaration of TSMRemoteFiles component which allow to load a list of files from remote web-server.

Components

[TSMRemoteFiles](#)

See also

[SubFolders](#)

[Wildcard](#)

[FileList](#)

Directory property example

```
SMRemoteFiles.Directory := edDirectory.Text;  
SMRemoteFiles.Wildcard := '*.zip';  
SMRemoteFiles.SubFolders := True;
```

```
if SMRemoteFiles.Execute then  
lbFiles.Items.Assign(SMRemoteFiles.FileList);
```

See also

[SubFolders](#)

[Wildcard](#)

[Directory](#).

See also

[Directory](#)

[Wildcard](#)

[FileList](#)

See also

[SubFolders](#)

[Directory](#).

[FileList](#)

Properties

► Run-time only ■ Key properties

■ [Password](#)

■ [Port](#)

■ [Server](#)

■ [UseProxy](#)

■ [User](#)

Methods

- Key methods

~~Assign~~[linkDelphi=Assign_Method](#)}

- [FillProxyInfo](#)

See also

[Server](#)

[Port](#)

[UseProxy](#)

[User](#)

[FillProxyInfo](#)

Password property example

```
SMIWebUpdater.Proxy.UseServer := True;  
SMIWebUpdater.Proxy.Server := '192.190.01.106';  
SMIWebUpdater.Proxy.User := 'peter';  
SMIWebUpdater.Proxy.Password := '123';
```

See also

[Server](#)

[UseProxy.](#)

[User](#)

[Password](#)

[FillProxyInfo](#)

Port property example

```
SMIWebUpdater.Proxy.UseServer := True;  
SMIWebUpdater.Proxy.Server := '192.190.01.106';  
SMIWebUpdater.Proxy.Port := 8081;
```

See also

[UseProxy.
Port](#)
[User
Password](#)

[FillProxyInfo](#)

Server property example

```
SMIWebUpdater.Proxy.UseServer := True;  
SMIWebUpdater.Proxy.Server := '192.190.01.106';  
SMIWebUpdater.Proxy.Port := 8081;
```

See also

[Server](#)

[Port](#)

[User](#)

[Password](#)

[FillProxyInfo](#)

UseProxy property example

```
SMIHTTPClient.Proxy.UseServer := True;  
SMIHTTPClient.Proxy.Server := '192.190.01.106';
```

See also

[Password](#)

[Server](#)

[Port](#)

[UseProxy](#)

[FillProxyInfo](#)

User property example

```
SMIWebUpdater.Proxy.UseServer := True;  
SMIWebUpdater.Proxy.Server := '192.190.01.106';  
SMIWebUpdater.Proxy.User := 'peter';  
SMIWebUpdater.Proxy.Password := '123';
```

Properties

▶ Run-time only ■ Key properties

▶ ■ [Activity](#)

▶ ■ [ShipmentInfo](#)

■ [TrackingNumber](#)

Methods

- Key methods

Create{linkDelphi=Create_Method}

Destroy{linkDelphi=Destroy_Method}

GetQuery{linkDelphi=GetQuery_Method}

AfterExecute{linkDelphi=AfterExecute_Method}

SMIFedEx unit

In this unit you will find a declaration of TSMFedExTracker component.

Components

[TSMFedExTracker](#)

See also

[ShipmentInfo](#)

Activity property example

```
for i := 0 to SMFedExTracker.Activity.Count-1 do
begin
{date-time of activity}
ShowMessage(GetColumn(SMFedExTracker.Activity[i], 0, '=') + ' ' +
GetColumn(SMFedExTracker.Activity[i], 1, '='));
{status}
ShowMessage(GetColumn(SMFedExTracker.Activity[i], 2, '='));
{location}
ShowMessage(GetColumn(SMFedExTracker.Activity[i], 3, '='));
{comments}
ShowMessage(GetColumn(SMFedExTracker.Activity[i], 4, '='));
end;
```


See also

[Activity](#).

ShipmentInfo property example

```
for i := 0 to SMFedExTracker.ShipmentInfo.Count-1 do  
begin  
  ShowMessage('Parameter: ' + GetColumn(SMFedExTracker.ShipmentInfo[i], 0, '='));  
  ShowMessage('Value: ' + GetColumn(SMFedExTracker.ShipmentInfo[i], 1, '='));  
end;
```

See also

[Activity](#).

[ShipmentInfo](#)

TrackingNumber property example

```
{define a package number}
SMFedExTracker.TrackingNumber := '040862810001923';

{track a package}
SMFedExTracker.Execute;

{show results}
for i := 0 to SMFedExTracker.ShipmentInfo.Count-1 do
begin
MemoResult.Add(GetColumn(SMFedExTracker.ShipmentInfo[i], 0, '=') + ' ' +
GetColumn(SMFedExTracker.ShipmentInfo[i], 1, '='));
end;
```

Properties

- ▶ Run-time only
- Key properties
 - ▶ ■ [Temperature](#)
 - [TrackingType](#)
 - [Zip](#)

Methods

- Key methods

Create{linkDelphi>Create_Method}

GetQuery{linkDelphi=GetQuery_Method}

AfterExecute{linkDelphi=AfterExecute_Method}

SMIWeath unit

In this unit you will find a declaration of TSMWeatherInfo component.

Components

[TSMWeatherInfo](#)

Types

[TSMWeatherTrackingType](#)

Temperature property example

```
SMWeatherInfo1.TrackingType := wttZip;  
SMWeatherInfo1.Zip := '02090';  
ShowMessage('Temperature is ' + IntToStr(SMWeatherInfo1.Temperature))
```


Properties

► Run-time only ■ Key properties

- [Description](#)
- [Enabled](#)
- [HelpContext](#)
- [Icon](#)
- [Picture](#)
- [Title](#)

Methods

- Key methods

- ~~Create~~{linkDelphi=Create_Method}

~~Destroy~~{linkDelphi=Destroy_Method}

~~Assign~~{linkDelphi=Assign_Method}

Title property example

```
SMWebUpdater1.StatusDialog.Enabled := True;  
SMWebUpdater1.StatusDialog.Title := 'Data file update';  
SMWebUpdater1.StatusDialog.Description := 'Wait until process is in progress...';  
SMWebUpdater1.StatusDialog.Icon.Assign(Application.Icon);  
SMWebUpdater1.StatusDialog.Picture.Assign(imgLogo.Picture);  
...  
SMWebUpdater1.Execute.
```

TSMIAction type

[See also](#)

Unit

[smiupdate](#)

Declaration

```
type TSMIAction = (iaIdle, iaConnecting, iaConnected,  
iaDisconnecting, iaDisconnected, iaFileSearch, iaFileFound,  
iaComparing, iaCompared, iaDownloading, iaDownloaded);
```

Description

This is a set of available values for Action property.

iaIdle component do nothing in wait mode

iaConnecting component started to try to connect to remote web server

iaConnected component connected to remote web server

iaDisconnecting component started to try to disconnect from remote web server

iaDisconnected component disconnected from remote web server

iaFileSearch component started to try to find a required file at remote web server

iaFileFound component found a required file at remote web server

iaComparing component started to compare a found remote file and defined locale file

iaCompared component compared a found remote file and defined locale file

iaDownloading component started to download a remote file

iaDownloaded remote file is downloaded

TSMIFileCompare type

[See also](#)

Unit

[smiupdate](#)

Declaration

```
type TSMIFileCompare = procedure (Sender: TObject; var Equal: Boolean); of object;
```

Description

When remote file is found and general information about this remote file is retrieved, component must have decide: to download a file or not.

To change this decision you can in OnFileCompare event.

If Equal parameter is False, then remote file will be not downloaded.

TSMIPProgress type

[See also](#)

Unit

[smiupdate](#)

Declaration

```
type TSMIPProgress = procedure (Sender: TObject; CurValue, MaxValue: Integer; var Abort: Boolean); of object;
```

Description

During file downloading you can control a process in the OnProgress event. Most popular task is to update a progressbar or cancel a downloading.

This event will be called after every block of retrieved information from web-server.

TSMIService type

[See also](#)

Unit

[smiupdate](#)

Declaration

```
type TSMIService = (isHTTP, isFTP, isNetwork);
```

Description

This is a set of available values for Service property.

isHTTP component must use the HTTP-protocol for update/download

isFTP component must use the FTP-protocol for update/download

isNetwork component must work with local network (shared directory) for update/download

TSMIStatus type

[See also](#)

Unit

[smiupdate](#)

Declaration

```
type TSMIStatus = (isInProgress, isFinished, isFailed, isSkipped,  
isNotFound);
```

Description

This is a set of available values for Status property.

isInProgress update/download process is in progress

isFinished update/download process is finished succesfully

isFailed update/download process is failed with some error

isSkipped update process is finished but remote and local files are identical

isNotFound update process is stopped because remote file is not found

TSMIShowMode type

[See also](#)

Unit

[smidirect](#)

Declaration

```
type TSMIShowMode = (omAlways, omNever, omIfChanged);
```

Description

This is a set of available values for ShowMode property.

omAlways dialog will be displayed in any situation. No matter news are changed or not

omNever dialog will be not displayed. Only news will be download in local copy

omIfChanged dialog will be displayed only if some changed news are found

TSMUPSStatus type

[See also](#)

Unit

[SMIUPS](#)

Declaration

```
type TSMUPSStatus = (ssTrackingNumber, ssLastActivity,  
ssLastStatus, ssServiceType, ssWeight, ssWeightUnits, ssSignedBy,  
ssLocation);
```

Description

This set defines a list of available information which can be extracted from UPS Online as answer to package tracking.

ssTrackingNumber to read a number of package

ssLastActivity to read a date of last activity by package

ssLastStatus to read a last status of package

ssServiceType to read a service type

ssWeight to read a weight of package

ssWeightUnits to read a weight units of package

ssSignedBy to read a name of person who signed a last step

ssLocation to read a location of package

TSMUPSTrackingType type

[See also](#)

Unit

[SMIUPS](#)

Declaration

```
type TSMUPSTrackingType = (upsCurrentStatus, upsFullTracking);
```

Description

This set defines a list of available types for UPS tracking:

upsCurrentStatus to retrieve a short information about current state of package

upsFullTracking to retrieve a detailed information about every step in package deployment

TSMIExtract type

[See also](#)

Unit

[smixtract](#)

Declaration

```
type TSMIExtract = (ieMailTo, ieHTTP, itHTTPS, itFTP, itFile,  
itTelnet, itNews, itGopher, itWais);
```

Description

This set defines a list of available prefixes for data extraction. Every value define a kind of data.

For example, ieMailTo allow to extract a list of published email addresses from url, ieHTTP will extract a list of files etc

TSMIHTTPAction type

[See also](#)

Unit

[SMIHTTP](#)

Declaration

```
type TSMIHTTPAction = (haHead, haGet, haPost);
```

Description

This set defines a list of available requests which could be sent to server.

Any http-server supports HEAD, GET and POST action.

TSMWeatherTrackingType type

[See also](#)

Unit

[SMIWeath](#)

Declaration

```
type TSMWeatherTrackingType = (wttZip);
```

Description

This enumerated type describe a list of available values for temperature tracking

See also

[Action](#)

[OnActionChange](#)

See also

[RemoteFileName](#)

[RemoteFileCreated](#)

[RemoteFileSize](#)

[OnFileCompare](#)

See also

[OnProgress](#)

See also

[Service](#)

See also

[Status](#)

[OnStatusChange](#)

See also

[ShowMode](#)

[Show](#)

[ShowModal](#)

See also

[GetUPSStatus](#)

[TSMTrackingType](#)

See also

[TrackingType](#)
[TSMUPSStatus](#)

See also

[Extract](#)

[ExtractedInfo](#)

See also

[Action](#)

[Request](#)

See also

[TrackingType](#)