

Socket Driver Contents

Socket Driver

Socket Utilities

Socket Driver

A socket is an endpoint in a bi-directional communication channel. The other end can be in the same task (not usually very interesting), in another task on the same machine, or in a task on another machine that is accessed through a TCP/IP connection.

The foreign family 16! : support sockets under NT and Win95. The Socket Driver is not available on the Macintosh. The Socket Driver works with 32 bit TCP/IP and does not work with a 16 bit TCP/IP stack.

File system\main\socket.ijs contains cover functions for the new Socket Driver. To load, enter:

```
load 'socket'
```

Directory system\examples\socket contains examples of using the Socket Driver to communicate between two J sessions. See system\examples\socket\socket.txt for details.

To use sockets you need to have TCP/IP support configured. The following is an excerpt from a FAQ:

[Q: How do I set up my computer for a TCP/IP network?]

1. In Control Panel, double-click the Network icon.
 2. On the Configuration tab, click Add, and then double-click Protocol.
 3. Click Microsoft, and then click TCP/IP
- After it is installed, click TCP/IP on the Configuration tab of Network properties, and then click Properties. Configure your protocol per instructions from your system administrator.

The Socket Driver is a very direct mapping onto the Windows Sockets 1.1 API. interface. General documentation on sockets and the Windows API is relevant and useful for complete understanding of how best to make use of sockets. Web sites www.stardust.com and www.sockets.com have lots of relevant information for a serious socket application developer. In particular, you might want to download the complete Windows Sockets 1.1 Specification from:

www.startdust.com/wsresource/winsock1/ws1docs.html.

There are labs (Studio|Labs) on sockets.

Socket Utilities

Script `system\main\socket.ijs` has definitions for working with sockets.

The first element of the result of all socket verbs is a result code. It is 0 if there was no error, or it is an error number. Utility `sderror` returns the text name of the error number.

Some socket verbs take integer constants as arguments and some of these constants have been given names in `socket.ijs`. For example `SOCK_STREAM` is 1.

A socket can be blocking or non-blocking. A verb such as `sdrecv` will hang on a blocking socket until it can complete. On a non-blocking socket, the `sdrecv` returns immediately with a result code of `EWOULDBLOCK 10035`. In Windows the use of non-blocking sockets is recommend.

A socket created with `ssocket` is a blocking socket. The verb `sdiocctl` can make it non-blocking. In Windows it is recommended to use `sdasync` to make the socket non-blocking as this also marks the socket so that events are notified to J by running sentence `socket_handler ' ' .` This is similar to window events and the `wdhandler ' ' .` sentence.

Verb `sdselect` returns information about the state of a socket and indicates if it has data to read, is ready for writing, or has had an error.

Addresses used with sockets consist of 3 parts: family, address, port. The first is an integer which indicates the type of address. Currently this is always `AF_INET` (address family internet). The second part is a string that is a series of 1 to 4 numbers separated by dots. The third part is an integer port.

```
ssocket family , type , protocol  
ssocket ' '
```

Creates a new socket. A socket is identified by an integer. The family must be `AF_INET` as defined in `sockets.ijs`. The type can be any of the `SOCK_` values, but is usually `SOCK_STREAM`. The protocol must be 0. The result is a socket number that can be used as the socket argument to other verbs. The `' '` argument is equivalent to `AF_INET, SOCK_STREAM, 0`

```
sdrecv socket , count , flags
```

Receives data from a socket. The count is the maximum amount of data that will be received. The flags are from the `MSG_` values and is usually 0. The result is a boxed list. The first element is the result code and the second is the data. There may be less data received than in count.

If the socket is blocking and there is no data, the verb will hang until there is data.

If the socket is non-blocking and there is no data, it immediately returns result code `EWOULDBLOCK 10035`.

`sdiocctl` can be used to see how much data is available for a socket.

If the socket at the other end is closed, then the socket will be in the `sdselect` ready-to-read list and an `sdrecv` will immediately receive 0 characters with no error.

If `sdasync` has been done for a socket, then the `socket_handler''` is run whenever new data is available.

```
sdrecv sk , 1000 , 0
```

```
data sdsend socket , flags
```

The left argument is the data to send. The flags are from the `MSG_` values and is usually 0.

Blocking and non-blocking sockets work with `sdsend` in a similar manner to `sdrecv`.

The second element of the result indicates how many characters were actually sent. This may be less than was requested and you need to call `sdsend` again to send the remaining data.

```
'testing 1 2 3' sdsend sk , 0
```

```
sdrecvfrom socket , count , flags
```

Similar to `sdrecv` except it is used with a `SOCK_DGRAM` socket. The result has additional elements that give the address of the data source.

```
data sdsendto socket ; flags ; family ; address ; port
```

Similar to `sdsend` except it is typically used with a `SOCK_DGRAM` socket and the argument includes the address to send the data to.

```
'test' sdsend sk ; 0 ; AF_INET ; '127.0.0.1' ; 800
```

```
sdclose socket
```

Close a socket.

```
sdconnect socket , family , address , port
```

Connect the socket to the socket indicated by the address.

An `sdconnect` on a blocking socket will hang until it completes and will either return a 0 result code indicating success, or an error code.

An `sdconnect` on a non-blocking socket returns immediately with `EWOULDBLOCK` 10035. The system will try to complete the connection asynchronously. If it is successful, the socket will be marked ready for writing in `sdselect`. If the connection fails the socket will be marked in error in `sdselect`.

```
sdconnect sk ; AF_INET ; '127.0.0.1' ; 800
```

```
sdbind socket , family , address , port
```

Bind a socket to an address. The address can be `''` if the socket will be used to listen for connects to any address on the machine. If the port number is 0, the system will assign a port (which can be queried with `sdgetsockname`).

A bind is usually done with a socket that will listen for connections.

```
sdbind sk ; AF_INET ; '' ; 800 NB. any connections to 800
```

```
sdlisten socket , number
```

Set the socket to listen for connections. A bind must have been done. The number is the limit to queued connections. The host typically forces this limit to be between 1 and 5.

When a connection is made the socket is marked in `sdselect` as ready for reading. When it is ready `sdaccept` should be done.

```
sdaccept socket
```

When a listening socket is marked as ready for reading in `sdselect`, then an accept can be done to create a new socket for this end of the channel. The new socket is a clone of the listening socket and has all its attributes. In particular, if the listening socket is non-blocking or has been marked with `sdasync`, then the new socket is as well. The result is the result code and the new socket.

```
sdselect read ; write ; error ; timeout  
sdselect ''
```

The argument is a 4 element list. The first is a list of sockets to check for ready-to-read, the second is a list to check for ready-to-write, and the third is a list to check for errors. The last element is a timeout value in milliseconds. If it is 0, the select is non-blocking and returns immediately. If the timeout is not 0, it will return as soon as there is a socket to report on, but will not wait longer than the timeout value.

An empty argument checks all sockets for all conditions with a timeout of 0.

The result has a result code and 3 socket lists. The first is the list of ready-to-read sockets. The second is a list of ready-to-write sockets. The last is a list of sockets that had an error.

Ready-to-read sockets are sockets with data available for an `sdrecv` or listening sockets with an incoming connection

```
sdgetsockopt socket , option_level , option_name  
Returns the value of a socket option.
```

```
sdgetsockopt sk , SOL_SOCKET , SO_DEBUG  
sdgetsockopt sk , SOL_SOCKET , SO_LINGER
```

```
sdsetsockopt socket , option_level , option_name , value...  
Set the value of a socket option.
```

```
sdsetsockopt sk , SOL_SOCKET , SO_DEBUG , 1  
sdsetsockopt sk , SOL_SOCKET , SO_LINGER , 1 , 66
```

sdioctl socket , option , value
Read or write socket control information.

```
sdioctl sk , FIONBIO , 0 NB. set blocking
sdioctl sk , FIONBIO , 1 NB. set non-blocking
sdioctl sk , FIONREAD, 0 NB. count of data ready to read
```

sdgethostname ''
Returns host name.

sdgetpeername socket
Returns address of socket this socket is connected to.

sdgetsockname socket
Return address of this socket.

sdgethostbyname name
Returns an address from a name.

```
sdgethostbyname 'localhost'
+--+-----+
|0|2|127.0.0.1|
+--+-----+
sdgethostbyname >1{sdgethostname ''
+--+-----+
|0|2|204.92.48.126|
+--+-----+
sdgethostbyname 'www.jsoftware.com'
+--+-----+
|0|2|198.53.145.167|
+--+-----+
```

sdgethostbyaddr AF_INET , address_name
Returns a name from an address.

```
sdgethostbyaddr 2 ; '127.0.0.1'
+--+-----+
|0|localhost|
+--+-----+
```

sdgetsockets ''
Return result code and all socket numbers.

sdwsaasync socket
Make a socket non-blocking and cause the system to run sentence
socket_handler '' whenever the state of the socket has changed.

sdcleanup ''
Close all sockets and release all socket resources.