

## Socket Interface Programming Requirements

The Socket Interface is based on our current RPC interface and works in a similar fashion.

Request data is sent to our host from your client using sockets. If the security information in your request is valid, then you will receive a response that contains your requested data, acknowledgement of data you submitted to Infosolutions, or error information describing what failed in the transaction.

If the security information is not valid, you will receive no response at all. We do this to provide no help of any kind for criminals attempting to hack into our systems.

The request data is made up of at least two parts, a **Socket Interface Header** and one or more **Infosolutions Messages**. The **Socket Interface Header** is **always** clear text, never encrypted or compressed by the client (of course secured connections may encrypt and decrypt the data automatically). The **Infosolutions Messages** will be compressed and/or encrypted based on flags set in the **Socket Interface Header**.

The **Socket Interface Header** is always 256 bytes in length. All information is represented by ASCII text to avoid concerns with data order differences between computers.

Name	Byte(s)	Length	Description
Length	0-11	12	Total length of the request including the header
Vendor ID	12-14	3	Assigned Vendor ID
Encryption	15	1	Encryption flag, 0=none, 1=Des2
Compression	16	1	Compression flag, 0=none, 1=PK
User ID	17-24	8	Provider's Submitter ID
Transaction ID	25-32	8	Unused, but returned with response. Can be used to match responses with requests.
Version	33-35	3	Socket Header/Interface version = 000
Unused	36-255	220	Available for future development

Table 1. Socket Interface Header

The **Infosolutions Messages** are exactly as described in the **Infosolutions Interface Manual**.

In operation, our socket's server will wait for a connection to be established. We will then hand that connection off to it's own child process and continue looking for new connections. The child process will look for 256 bytes of data. If we time out before all this data is received in any step or if the security information is incorrect, we will break the connection and the child will terminate. If the 256 bytes are received and prove to be a valid **Socket Interface Header**, we will attempt to receive the amount of data specified in the **Socket Interface Header**, decrypt and/or decompress it as specified in the **Socket Interface Header**, and pass the data on to be processed. When we receive the response (or generate an error message), we will compress and/or encrypt if specified, transmit that data with it's own **Socket Interface Header** (a copy of the client header, except length matches the returned data).