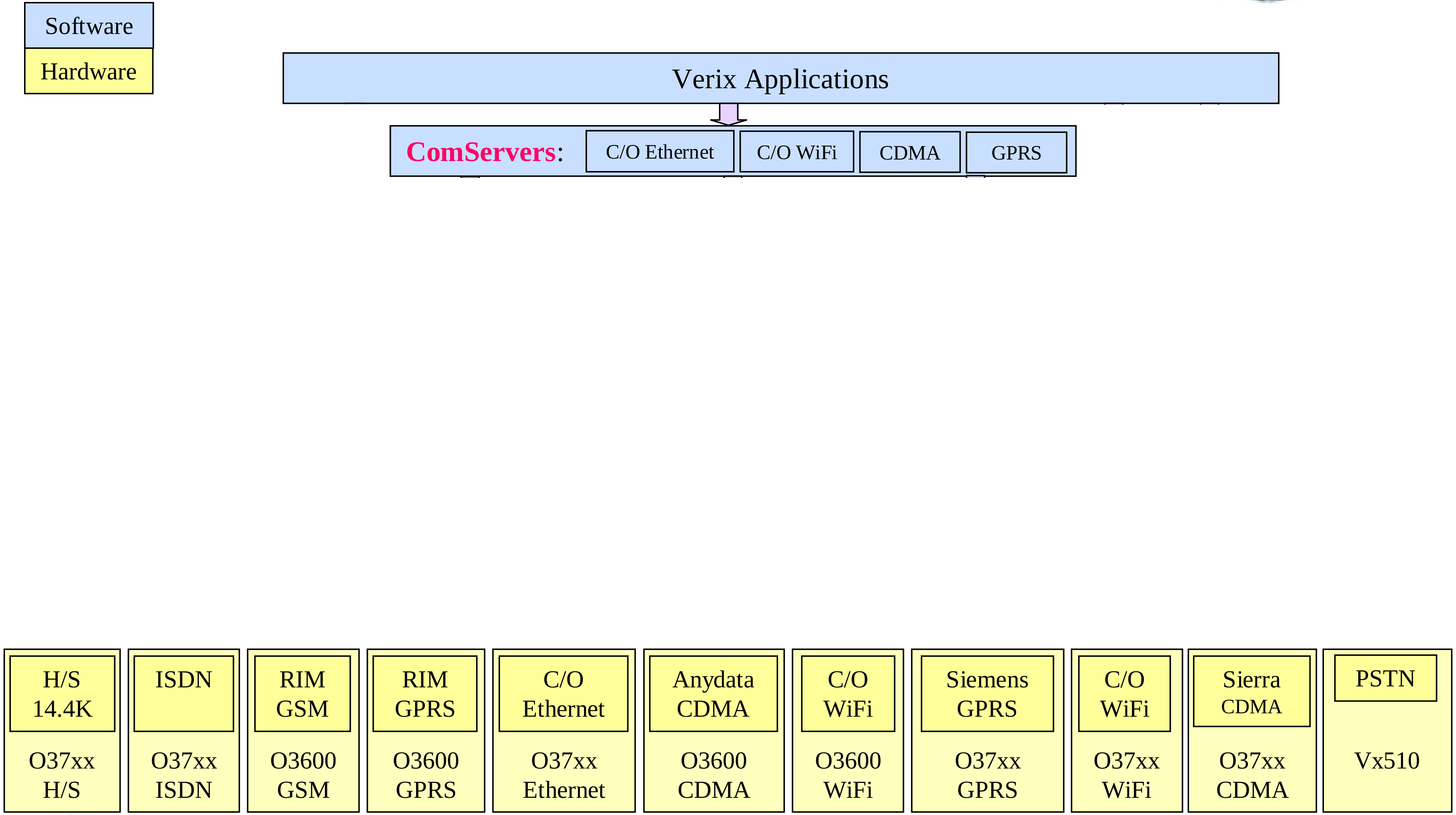
VCS

Verix Communication Server

VCS Library



Software

Hardware

Verix Applications

ComServers: C/O Ethernet | C/O WiFi | CDMA | GPRS

H/S 14.4K	ISDN	RIM GSM	RIM GPRS	C/O Ethernet	Anydata CDMA	C/O WiFi	Siemens GPRS	C/O WiFi	Sierra CDMA	PSTN
O37xx H/S	O37xx ISDN	O3600 GSM	O3600 GPRS	O37xx Ethernet	O3600 CDMA	O3600 WiFi	O37xx GPRS	O37xx WiFi	O37xx CDMA	Vx510

What is CommServer

§ The Communication Server Framework uses the VMAC architecture to create a client/server solution; where a communication server task is responsible for all communications from the session layer down

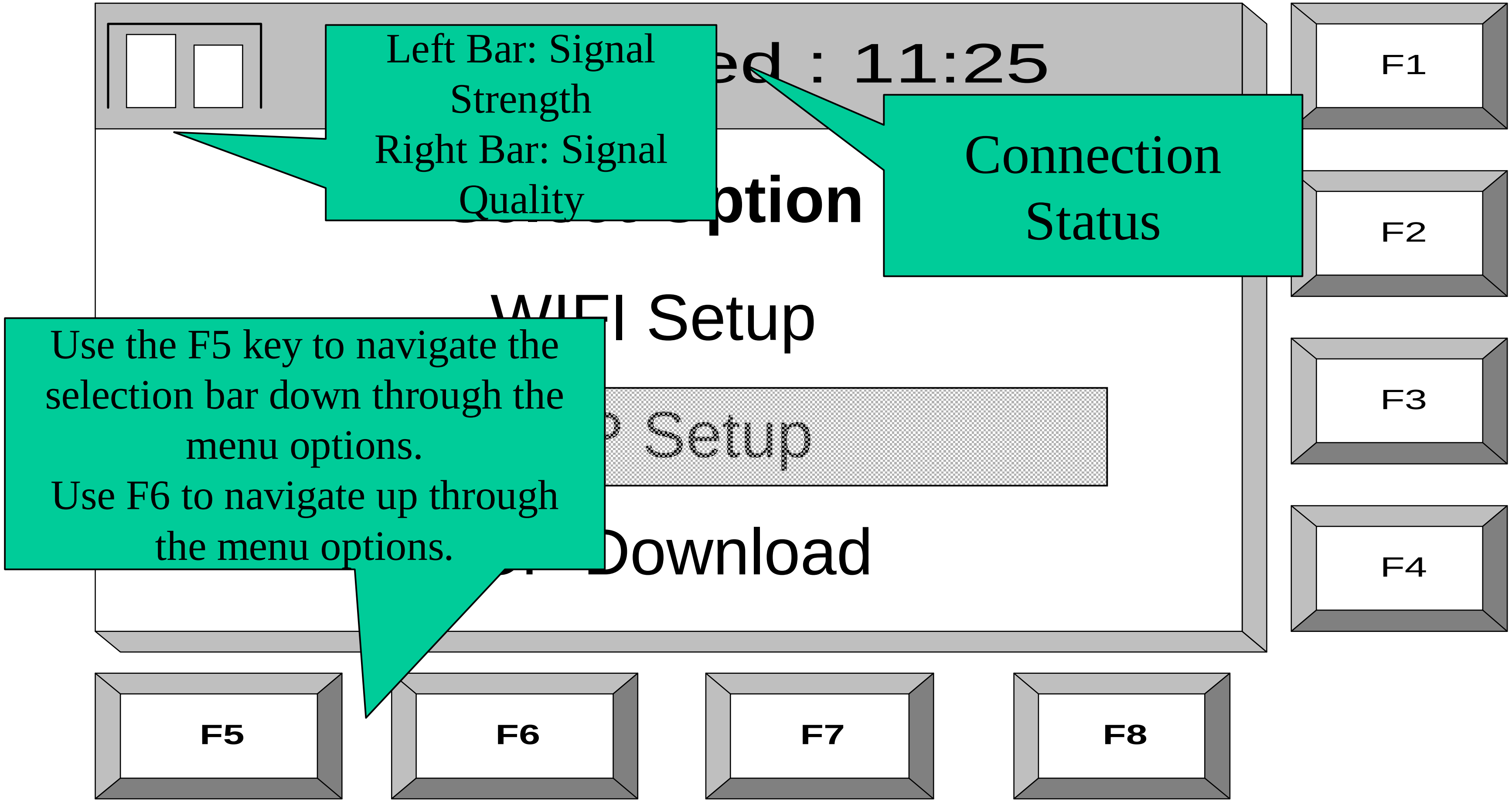
§ An application communicates with server task using VMAC EESL messages

§ Each server task is developed according to the Verix Communication Server (VCS) specification

Advantages of CommServer

- Concurrent application operation:
  - Transaction Data Entry
  - SSL Connection
- Multiple concurrent communication sessions from multiple applications
- Application independence from communication mediums
- User Interface
  - Configuration
  - Status
  - Troubleshooting
- Shared Application Service
  - Single instance of Library (TCP/IP)
  - Smaller client application foot-print

UI: Main Menu Navigation

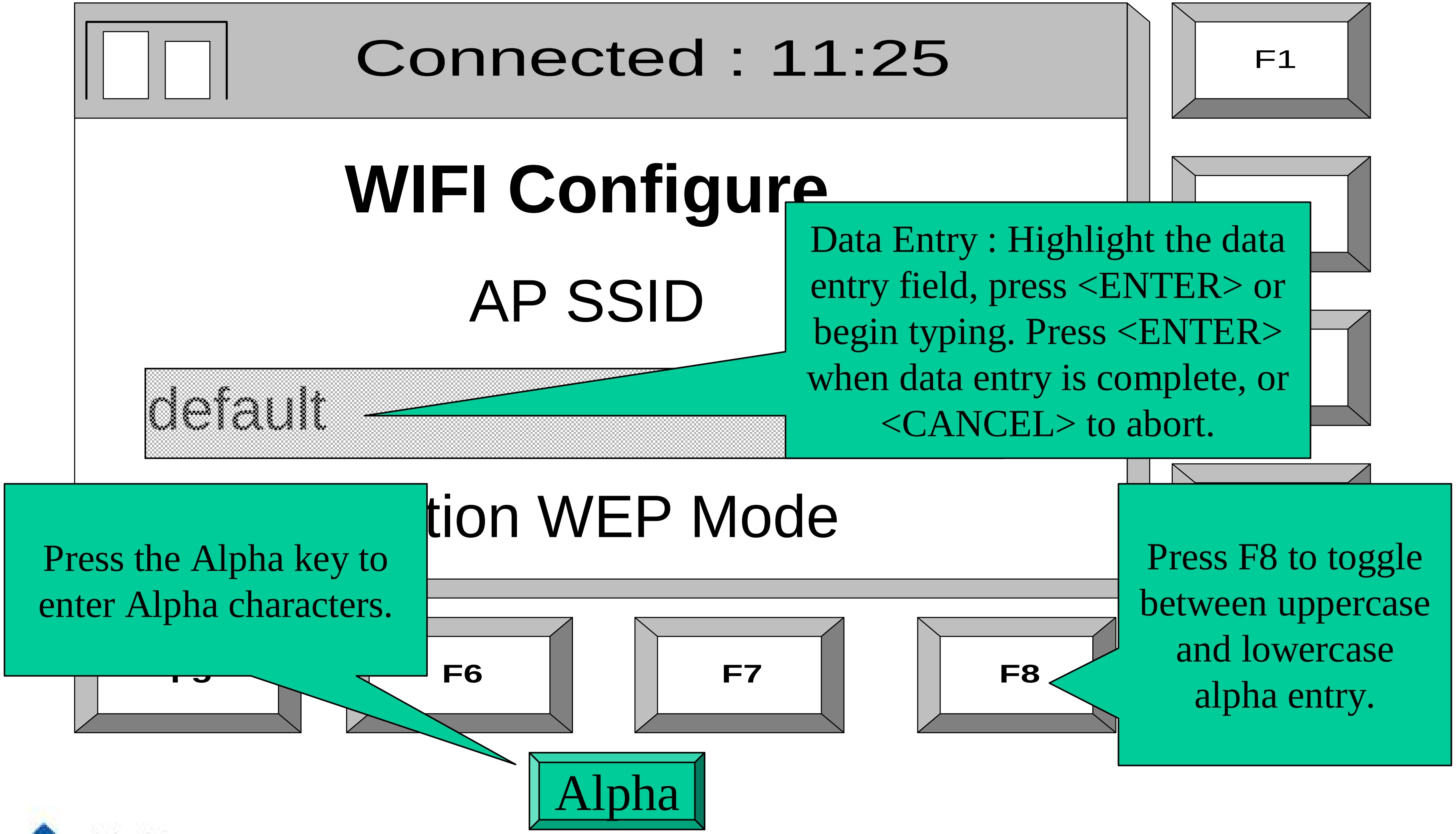


Left Bar: Signal Strength
Right Bar: Signal Quality

Connection Status

Use the F5 key to navigate the selection bar down through the menu options.
Use F6 to navigate up through the menu options.

UI: Data Entry



Connected : 11:25

WIFI Configure

AP SSID

default

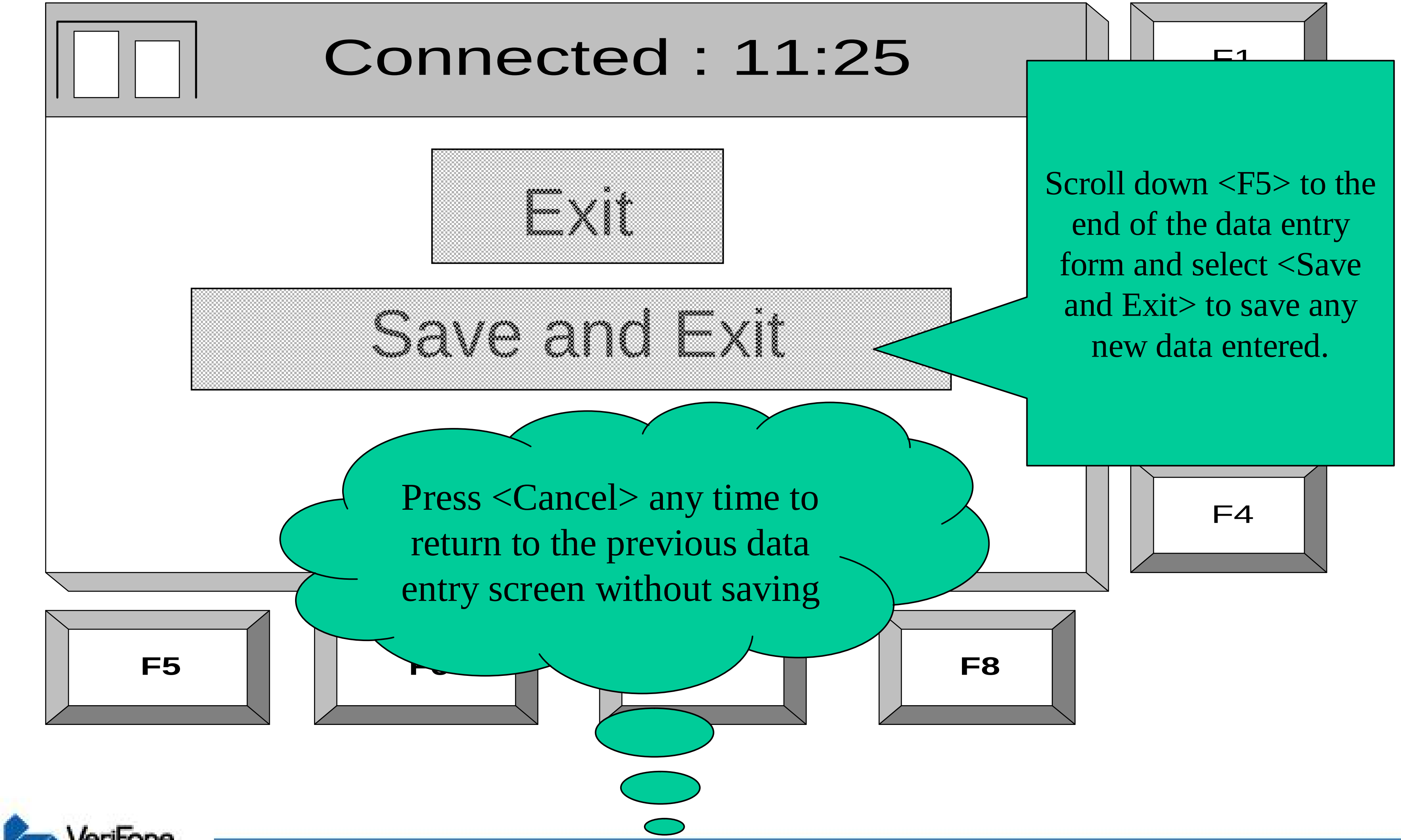
tion WEP Mode

Data Entry : Highlight the data entry field, press <ENTER> or begin typing. Press <ENTER> when data entry is complete, or <CANCEL> to abort.

Press the Alpha key to enter Alpha characters.

Press F8 to toggle between uppercase and lowercase alpha entry.

UI: Saving Configuration Data



Scroll down <F5> to the end of the data entry form and select <Save and Exit> to save any new data entered.

Press <Cancel> any time to return to the previous data entry screen without saving

CommServer Instantiations

- A CommServer instantiation is a Verix executable which follows the CommServer message interface specification

CommServer Instantiations:
For O37xx IOM’s

- CommServer instantiations are available for the following Omni 37xx wireless IOM’s:
  - Siemens GPRS
  - Sierra Wireless CDMA
  - Connect One WIFI

CommServer Instantiations: For Vx610

- CommServer instantiations are available for the following Vx610 terminals:
  - Siemens GPRS
  - Sierra Wireless CDMA
  - Connect One WIFI

CommServer Features

- CommServer supports the following features:
  - TCPIP/SSL Application downloads
    - Download Configuration through Config.sys or User Interface
    - CommServer can be executed via OS system mode if TCPIP download option is selected
  - Main Menu Ping option allows user to ping an IP address or URL

Wireless CommServer Features

- Signal strength monitoring and display
- Connection state monitoring and display
- Automatic re-connect:
  - If CommServer detects the data connection (CDMA or GPRS) has been dropped (DCD is detected low), CommServer will immediately attempt a new PPP connection to the data network
  - The re-connect will only take place if CommServer is not in focus

Vx610 CommServer Features

- Power Save Compliance
  - CommServer is responsible for setting the wireless device to low power mode
  - CommServer will not enter a “Ready for Power Save” state while it receives messages from a client application

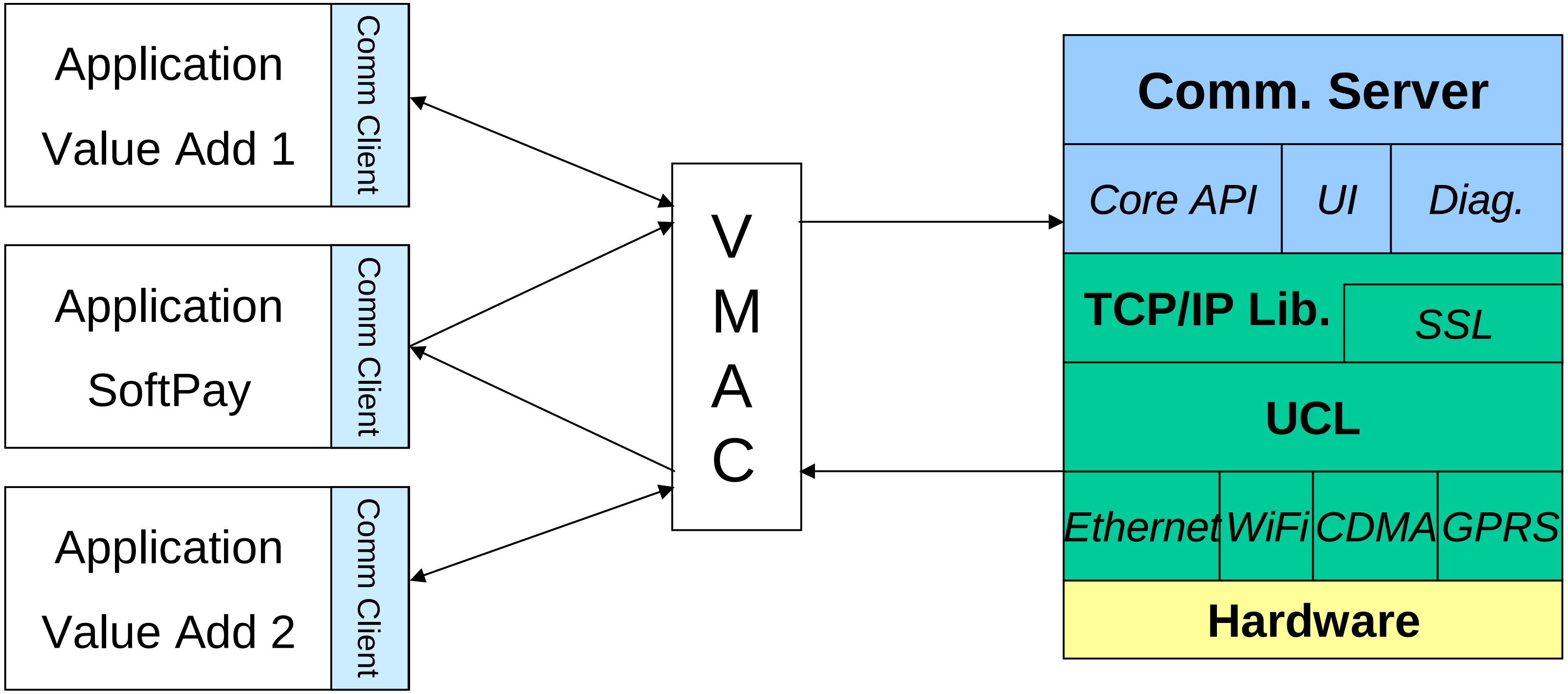
CommServer Device Configuration

- Password protected
- Device Specific Configuration
  - WIFI
    - Channel
    - SSID
    - WEP Mode
    - WEP Key index
    - 4 WEP Keys
  - GPRS
    - Dial String
    - APN
  - CDMA
    - Dial String

TCP/IP Configuration

- Password protected
- CommServer will attempt DHCP or IPCP if the IP fields are left as 0.0.0.0, or are blank
- TCP/IP Configuration fields:
  - IP address
  - Subnet mask
  - Gateway address
  - 2 DNS addresses

Communication Server Architecture



37xx CommServer COM3 Ownership

- A CommServer instantiation should own COM3 by default, I.e. have a COMM_3_AVAILABLE_EVENT entry in its device mapping table

- A CommServer instantiation should own COM3 by default, so that Device Manager will request COM3 from CommServer if another application request COM3 with high priority

Vx610 CommServer Device Ownership

- On the Vx610 devices, COM2 is assigned to the Radio and COM3 to the dial Modem
- Application use of COM3 and COM2 is mutually exclusive

CommServer Message Interface

- The crux of the CommServer definition is the message interface
- All message records use the Flexi Record format
    - Only raw data messages do not use the Flexi Record format
    - Raw data messages have a size limitation of 500 bytes

CommServer Client Best Practices

- Check error values when calling EESL_Send_Event()
  - Resend messages If and error occurs (+- once every 500ms).
- Use ushInitStandardFlexi() to initialize Flexi Record field types before adding or retrieving fields.
- Use typed Flexi Record functions to retrieve fields:
  - shVarGetUnsignedInt()
  - shVarAddUnsignedInt()
- Check flexi error values when retrieving fields from a flexi record.
- Initialize variables before they are passed as parameters to Flexi Record functions.

VCS_EVT_INIT_REQ

- Initializes as session with the server task.

VCS_EVT_INIT_RESP

- Returns the session handle

Flexi Record Field	Field type	Description
VCS_FLD_SESS_ERROR	Int	Generic error values (see Appendix A)
VCS_FLD_SESS_NATIVE	Int	Implementation specific error value
VCS_FLD_SESS_HANDLE	Int	Non-negative handle to be presented in all future events sent to the server

VCS_EVT_DEINIT_REQ

- Closes a client/server session

Flexi Record Field	Field type	Description
VCS_FLD_SESS_HANDLE	Int	Non-negative handle to be presented in all future events sent to the server

VCS_EVT_DEINIT_RESP

- Response from the server indicating the success or failure of the de-init process

Flexi Record Field	Field type	Description
VCS_FLD_SESS_ERROR	Int	Generic error values (see Appendix A)
VCS_FLD_SESS_NATIVE	Int	Implementation specific error value

VCS_EVT_CONN_REQ

- Request a Socket Connection

Flexi Record Field	Field type	Description
VCS_FLD_CONN_HOSTSSL	Int	Flag indicating whether SSL is supported: 1 - SSL supported 0 - SSL not supported
VCS_FLD_CONN_CLNTAUTH	Int	Flag indicating whether client authentication is supported: Field is required only if SSL is enabled (see above field). Can contain 1 or 0.
VCS_FLD_CONN_SSLCERT	String	SSL certificate file path- does not contain GID information. Only required if client authentication flag is set.
VCS_FLD_CONN_PVTKEY	String	SSL private key Only requires if client authentication flag is set.
VCS_FLD_CONN_URL	String	Internet URL or an IP address
VCS_FLD_CONN_PORT	Int	TCP/IP port number
VCS_FLD_CONN_APPGROUP	Int	Implementation specific error value
VCS_FLD_SESS_HANDLE	Int	Non-negative handle to be presented in all future events sent to the server

VCS_EVT_CONN_RESP

- Result of a socket connection request

Flexi Record Field	Field type	Description
VCS_FLD_SESS_ERROR	Int	Generic error values (see Appendix A)
VCS_FLD_SESS_NATIVE	Int	Implementation specific error value

VCS_EVT_DISC_REQ

- A socket disconnect request

Flexi Record Field	Field type	Description
VCS_FLD_SESS_HANDLE	Int	Non-negative handle to be presented in all future events sent to the server

VCS_EVT_DISC_RESP

- The server socket disconnect response

Flexi Record Field	Field type	Description
VCS_FLD_SESS_ERROR	Int	Generic error values (see Appendix A)
VCS_FLD_SESS_NATIVE	Int	Implementation specific error value

VCS_EVT_SEND_REQ

- Client data send request

Flexi Record Field	Field type	Description
VCS_FLD_SEND_BUFSIZE	Int	Size of the data to be sent from the communications server to the host
VCS_FLD_SESS_HANDLE	Int	Non-negative handle to be presented in all future events sent to the server

VCS_EVT_SEND_RESP

- Server data send response

Flexi Record Field	Field type	Description
VCS_FLD_SESS_ERROR	Int	Generic error values (see Appendix A)
VCS_FLD_SESS_NATIVE	Int	Implementation specific error value

VCS_EVT_RECV_REQ

- Client receive request

Flexi Record Field	Field type	Description
VCS_FLD_RECV_TIMEOUT	Int	Timeout in seconds that the server will attempt to read data
VCS_FLD_RECV_BUFSIZE	Int	Maximum number of bytes to be read
VCS_FLD_SESS_HANDLE	Int	Non-negative handle to be presented in all future events sent to the server

VCS_EVT_RECV_RESP

- Client receive response

Flexi Record Field	Field type	Description
VCS_FLD_SESS_ERROR	Int	Generic error values (see Appendix A)
VCS_FLD_SESS_NATIVE	Int	Implementation specific error value
VCS_FLD_RECV_BUFSIZE	Int	Size of data read by the server

VCS_EVT_DATA_RAW

- The data portion of this message is not formatted as a Flexi Record.

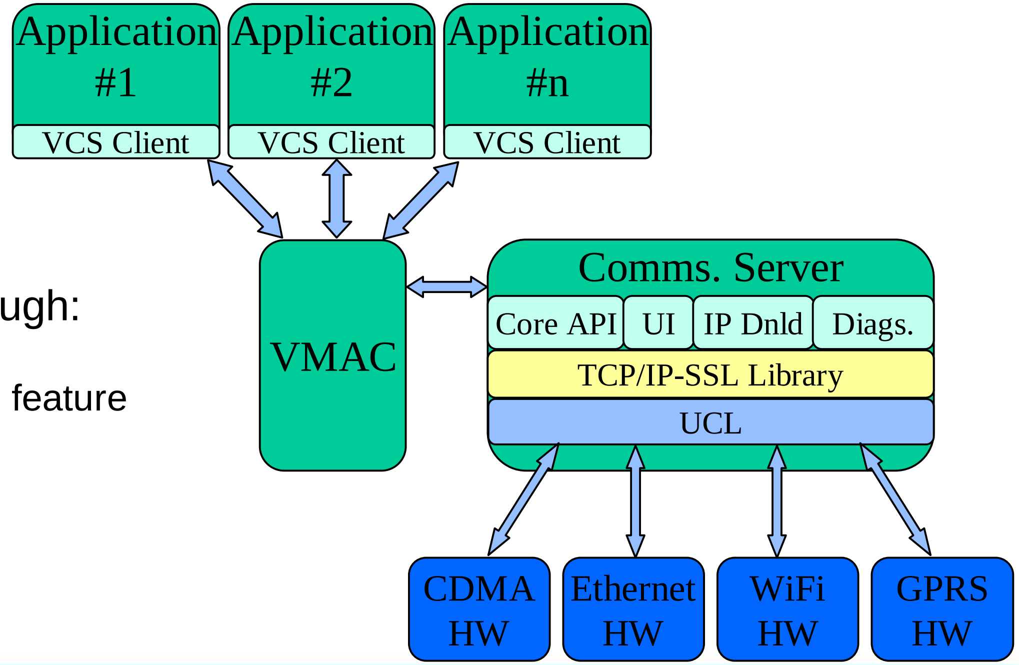
VCS_EVT_STATUS_REQ
VCS_EVT_STATUS_RESP

- Is sent by the client to request communication status data

Flexi Record Field	Field type	Description
VCS_FLD_STATUS_IDS	Buffer	Contains an array of integers. Each integer is a field ID of a communication status element that will be returned to the client
VCS_FLD_SESS_HANDLE	Int	Non-negative handle to be presented in all future events sent to the server

CommServer Summary

- VMAC based Client / Server solution for advanced communications
  - Applications communicate with server through VMAC message interface
- Server:
  - Provides shared communication services for all VCS enabled applications
  - Handles all communication tasks up to session layer
  - With or without SSL support
  - Components include:
    - Core APIs
    - Device Configuration UI
    - Device Status and Diagnostics
    - SysMode IP Download Module
  - Optimised memory footprint through:
    - UCL “Smart-Link” feature
    - TCP/IP-SSL library “Smart-Link” feature
  - VCS Files are preserved
    - On Vx610 and O3750
    - Similar to Opsys files.



Value Proposition Summary

- § Faster Time to Market
  - Application Independent Device User Interface
    - – Reduces Application Development time (and application size) offering: Configuration UI, Status UI, and Troubleshooting UI
  - Interoperability
    - – Delivers application independence from comm. device minimizing application re-certification when changing device
- § Faster Transactions
  - Through concurrent application operations, such as: transaction data entry in parallel to SSL connection establishment / resumption
- § Enhances Terminal Capabilities
  - Concurrent communication session for multiple applications.
- § Lowers HW Cost
  - Minimizes memory requirements in Multi-App use case through shared services
    - – Single instance of UCL
      - – Instead of per application use of UCL or alternative application specific device handling
    - – Single instance of TCP/IP stack
      - – Instead of per application use of TCP/IP library or alternative application specific stacks
    - – Single instance of SSL for SSL enabled instantiations
    - – Smaller client applications foot-print.