4D Internet Commands

Los comandos Internet de 4D

Prefacio

Los comandos Internet de 4D ("4D Internet Commands") ofrecen a los usuarios de 4D un robusto conjunto de herramientas de comunicación que permiten trabajar en todo tipo de red, local, nacional o mundial. La máxima expresión de esta explosión de conectividad se conoce coloquialmente como "Internet". En los últimos años se ha producido un enorme crecimiento en el número de personas y empresas con acceso a Internet. Como el volumen de personas con acceso a Internet aumenta, cada día hay más la necesidad de estar "en la red".

El conjunto de comandos ofrecidos por 4D Internet Commands da a los desarrolladores de base de datos 4D acceso a muchos elementos claves de Internet. Los comandos SMTP contienen herramientas para automatizar el envío de correo electrónico de una base de datos a una lista ilimitada de destinatarios. Del mismo modo, los comandos POP3 e IMAP permiten recuperar el correo de un número ilimitado de buzones para el almacenamiento en una base de datos, redireccionar, dar una respuesta automática o efectuar una búsqueda remota. Los comandos FTP permiten al usuario transferir archivos a/desde sistemas remotos u obtener listados de directorios de archivos en los volúmenes FTP. Los comandos TCP y UDP ofrecen a los desarrolladores herramientas de bajo nivel que les permiten ejecutar múltiples tareas relacionadas con Internet.

El protocolo SMTP (Simple Mail Transfer Protocol) es el principal protocolo de transferencia de correo usado en Internet. 4D Internet Commands permite a los usuarios crear y enviar rápidamente correo a través de un servidor SMTP. La creación y el envío de correo puede efectuarse con un solo comando. Si sus necesidades de entrega de correo son más complejas, cada aspecto del encabezado, cuerpo y archivos adjuntos del mensaje puede ser controlado. Ya que el correo Internet puede ser direccionado a las redes "privadas" tales como CompuServe, America Online, eWorld, etc. puede llegar a prácticamente cualquier persona con una cuenta de correo electrónico. El conjunto de comandos SMTP permite por ejemplo:

- el envío automatizado de estadísticas y de informes desde su base de datos
- la creación de una base de datos de reenvío automático de correo
- la gestión de una lista de correo (mailing list)
- la sincronizacion de base de datos remotas

Junto con sus comandos SMTP, 4D Internet commands también contiene comandos que se conectan a los servidores de correo electrónico POP3 (Post Office Protocol, versión 3) o IMAP (Internet Message Access Protocol) para la recuperación de mensajes electrónicos y de archivos adjuntos codificados. Dado que el conjunto de comandos SMTP, POP3 e IMAP cumplen con las normas MIME, múltiples mensajes y archivos adjuntos pueden descargarse y guardarse fácilmente.

Los comandos también permiten a los usuarios codificar los archivos adjuntos de diferentes maneras tales como: Binhex, Base64, AppleSingle, AppleDouble...

Los comandos FTP (File Transfer Protocol) ofrecen un mecanismo muy fácil de usar para comunicarse con un servidor FTP para enviar/recibir archivos de texto o binarios. Los comandos FTP permiten obtener la lista de directorios de archivos, facilitando el desarrollo de interfaces de navegación para volúmenes remotos. Los comandos FTP pueden utilizarse fácilmente en aplicaciones de búsqueda de documentos sin necesidad de "montar" los volúmenes remotos en el equipo cliente.

El protocolo TCP/IP (Transmission Control Protocol/Internet Protocol) es el protocolo principal utilizado para enviar y recibir datos a través de Internet. 4D Internet Commands contiene varios comandos para enviar y recibir paquetes TCP. Los comandos TCP ofrecen a los desarrolladores las herramientas necesarias para construir y controlar sus comunicaciones en Internet. Además, el comando *TCP_Open* permite la conexión a un servidor en modo seguro utilizando el protocolo SSL (Secured Socket Layer).

Algunos ejemplos son:

- Crear su propia interfaz telnet
- Ejecutar instrucciones en máquinas remotas
- Recuperar los documentos en la World Wide Web
- Efectuar búsquedas en las numerosas bases de datos en línea
- Manejar sincronizaciones de bases de datos con servidores remotos
- Seguimiento de paquetes Federal Express y UPS
- Conectarse a un servidor web por https.

Nota: para mayor flexibilidad, los comandos Internet de 4D permiten pasar directamente una referencia de conexión POP3, IMAP o FTP a los comandos de bajo nivel TCP y viceversa. Para obtener más información, consulte la sección **Rutinas de bajo nivel, Presentación**.

El protocolo UDP (User Datagram Protocol) es un protocolo desconectado que permite el envío y la recepción de datos de manera más rápida y más sencilla que TCP, pero con menor fiabilidad, ya que no permite la verificación de la entrega, comprobación de errores o la recuperación de datos entregados incorrectamente.

Instalación y requerimientos de software

Instalación

El plug-in "4D Internet Commands" se integra en 4D de la misma forma que los otros plug-ins.

4D Internet Commands está disponible cuando instala un producto 4D ya que el plug-in se instala automáticamente en la carpeta **PlugIns** de su aplicación.

Para mayor información sobre la instalación y configuración de plug-ins, consulte la Guía de instalación de la línea de productos 4D.

Requerimientos de software

4D Internet Commands requiere la misma configuración del sistema necesaria para 4D. Para obtener más información, consulte la *Guía de instalación 4D.*

Versión 64 bits

4D Internet Commands versión 64 bits debe utilizarse con 4D Server 64 bits:

- Windows: 4D Server 64 bits disponible a partir de la versión 12.1
- OS X: 4D Server 64 bits para OS X disponible a partir de la versión 14 R3

Acceso a la red

Para poder utilizar los comandos Internet de 4D, debe tener acceso a una red que soporte el protocolo TCP/IP.

TLS

4D Internet Commands le permite utilizar el protocolo seguro SSL con comandos para el envío de mensajes y conexión a los servidores de mensajería. No es necesaria una configuración especial.
Nota: la implementación de TLS en 4D Internet Commands utiliza el "método implícito".

Servidor de nombres de dominio

Para muchos de los comandos Internet de 4D, es necesario tener acceso a un servidor de nombres de dominios (Domain name server o DNS). Para mayor información, consulte su administrador de red.

Servidor de correo electrónico SMTP

Para enviar correo utilizando los comandos SMTP, es necesario que el remitente tenga acceso a un servidor de correo electrónico SMTP, el cual enviará el mensaje a un servidor de correo electrónico POP3.

Servidor de correo electrónico POP3

Para utilizar los comandos POP3, debe tener una cuenta en un servidor de correo electrónico POP3.

Servidor de correo electrónico IMAP

Para utilizar los comandos IMAP, debe tener una cuenta en un servidor de correo electrónico IMAP.

Glosario y terminología

Esta sección define muchas de las referencias realizadas a lo largo del manual. Las definiciones son simples y están dirigidas principalmente a aquellos no familiarizados con las referencias. La sección Terminología sobre "Formatos de los parámetros" ofrece información adicional sobre los parámetros de 4D Internet Commands.

NIC: "Network Information Center". Internet es en su mayoría una entidad no regulada. No existe una autoridad central o de control sobre su uso o crecimiento. Sin embargo, es necesario que un organismo único regule las atribuciones de nombres de dominio y de direcciones IP. El NIC es el grupo responsable de tales tareas administrativas.

RFC: "Request for Comments." La mayoría de los comandos Internet de 4D se basan en estándares definidos para controlar las comunicaciones en Internet. Las metodologías, descripciones y protocolos estándar utilizados en Internet se definen en los documentos conocidos como RFCs. **Anexo D: Información adicional** contiene referencias a algunos sitios web con enlaces a muchos de los documentos RFC. Aunque 4D Internet Commands simplifica la programación de los accesos a Internet, puede ser útil consultar algunos de estos documentos, especialmente si desea utilizar las rutinas de comunicación TCP de bajo nivel.

Direcciones TCP/IP, nombres de servidor y nombres de dominio: una dirección IP es una referencia a una máquina **específica** en algún lugar del mundo. La dirección IP tiene el formato de una cadena que contiene cuatro valores numéricos separados por puntos (es decir, "207.94.15.3"). Cada parte numérica de la dirección puede contener un valor entre cero y 255. Aplicando algunas funciones matemáticas a una dirección IP, su valor puede ajustarse a un número equivalente de tipo Entero largo, llamado **ip_LongInt** en este manual.

Para que un sitio (empresa, universidad, etc.) pueda conectarse a Internet, debe haber algunas garantías para asegurar que sus direcciones IP no entren en conflicto con otras máquinas en la red. Las empresas (y a menudo los particulares) registran su sitio con el **NIC** para obtener un **nombre de dominio**. **Los nombres de dominios** facilitan la identificación y lectura de direcciones de Internet. Los nombres de dominios son traducidos por el Domain Name System (DNS) a direcciones numéricas (números IP) utilizadas por la red. Este sistema permite un formato más fácil de leer como "www.4D.com" o "ftp.4D.com".

Nombre de dominio = "4D.com"

Host Name (Nombre de servidor)	**=**	**Dirección IP**	=	**ip_LongInt**
"www.4D.com"	=	"207.94.15.3"	=	-815919357

La correspondencia entre un **nombre de host** y su dirección IP se guarda en una base de datos llamada DNS (Domain Name System). Estos servidores se comunican entre sí para intercambiar los datos nuevos o modificados en las listas de nombres de dominios de todo el mundo. El panel de control TCP/IP ofrece un medio para "dirigir" su equipo a un DNS, que se encargará de traducir las referencias de nombres de dominio que utilice.

Es importante entender que todos los servidores de nombres de dominio tienen una dirección IP correspondiente. Sin embargo, no todas las direcciones IP tienen un servidor de nombres de dominios correspondiente. Asimismo, una dirección electrónica tal como "jsmith@4D.com" no hace referencia al equipo específico persona o dirección IP de esta persona. La dirección electrónica dirige la distribución de correo a la máquina con la dirección IP obtenida al convertir el dominio "4D.com". El correo se entrega al servidor POP3 que se ejecuta en esa máquina, que luego retendría el correo para su usuario llamado "jsmith".

Nombre de dominio: el nombre de dominio es una estructura de direccionamiento utilizada para la identificación y ubicación de ordenadores en Internet. Los nombres de dominios facilitan recordar las direcciones de Internet, que pueden ser traducidos por el Domain Name System (DNS) en direcciones numéricas (números IP) utilizados por la red. Un nombre de dominio es jerárquico y con frecuencia transmite información sobre el tipo de entidad que utiliza el nombre de dominio. Un nombre de dominio es simplemente una etiqueta que representa un dominio, que es un subconjunto del espacio de nombres de dominios total. Los nombres de dominio en el mismo nivel de la jerarquía deben ser únicos, por ejemplo, sólo puede haber un com al nivel superior de la jerarquía y un solo 4D.com en el siguiente nivel de la jerarquía. Si el nombre de su organización es "NombreEmpresa", podría registrar el nombre de dominio "NombreEmpresa.com" y su dirección de e-mail podría ser

"NombreUsuario@NombreEmpresa.com". Sus clientes también podrán acceder al sitio web de su organización, visitando "www.NombreEmpresa.com" con su navegador web.

Sistema de nombres de dominio (DNS): es una base de datos distribuida que almacena información que se utiliza para traducir nombres de dominios, fáciles de recordar y utilizar, en números IP, necesarios para ubicar los ordenadores en Internet . Los usuarios de todo el mundo conservan su parte de esta base de datos y los datos en cada porción de la base de datos están a disposición de todos los equipos y usuarios de Internet. El DNS consta de computadoras, archivos de datos, software y personas que trabajan juntos.

Codificación: la codificación convierte un archivo en un formato interpretable para todo tipo de sistema operativo (ASCII estándar). La forma más común de codificación es la codificación binaria hexadecimal (BinHex). La codificación BinHex es la opción de codificación por defecto para los archivos adjuntos que se agregan a los mensajes. Mientras que la codificación crea un nuevo archivo que es más grande que el original, convierte la parte datos (data fork) y la parte de recursos (resource fork) de un archivo en un documento de tipo texto que puede enviarse como un archivo adjunto. 4D Internet Commands soporta los métodos de codificación más comunes, incluyendo Binhex, Base64, AppleSingle, AppleDouble, UUEncode y MacBinary.

Cifrado: el cifrado se utiliza para codificar intencionalmente el contenido de los mensajes. Los mensajes son cifrados mediante un programa externo de cifrado PGP, con el único propósito de aumentar la privacidad de los mensajes. El texto cifrado debe ser descifrado antes de que pueda ser leído. 4D Internet Commands **NO** ofrece ningún medio para cifrar texto.

Compresión: se utiliza como medio para reducir el espacio del disco ocupado por un archivo. Para comprimir un archivo, puede utilizar una aplicación tal como StuffIt Deluxe ™ Compact Pro ™ o WinZip ™. Estos archivos deben descomprimirse utilizando la aplicación para devolver el archivo a su formato original. Las aplicaciones de compresión añaden generalmente un sufijo al nombre original del archivo. A continuación se presentan algunos sufijos comunes y sus respectivas aplicaciones.

NombreArchivo.SIT - aplicación Stuffit
NombreArchivo.CPT - aplicación Compact Pro
NombreArchivo.DD - aplicación Disk Doubler
NombreArchivo.ZIP - aplicación Winzip
NombreArchivo.SEA - Self Extracting Archive. Estos archivos son las aplicaciones Macintosh auto descomprimibles cuando el usuario hace doble clic en ellos, porque el código de descompresión se incluye. Debido a la adición de este código, los archivos auto extraíbles son por lo general más grandes que los creados como NombreArchivo.SIT o NombreArchivo.CPT. Sin embargo, como el usuario no necesita tener la aplicación de compresión, esta opción puede ser ventajosa para el usuario final.

Es importante recordar que una vez comprimido, un archivo todavía tiene que ser codificado antes de la transmisión para asegurar que el archivo sea transferido adecuadamente de una máquina a otra en su camino hacia su destino final.

Formato de los parámetros

Las descripciones a continuación ofrecen detalles sobre el significado y el formato de los parámetros más utilizados en este manual.

Parámetro	Tipo		Descripción
nomServidor	Cadena	→	Nombre del servidor (Ej: "www.nombredelaempresa.com") o Dirección IP (Ej: "204.118.90.2")
ip_EnteroLargo	Entero largo	→	Referencia de una dirección IP en forma de entero largo
direccionEmail	Texto	→	Ej: "jsmith@4d.com"
listaDirecciones	Texto	→	Ex: "jsmith@4d.com, jdupont@4d.fr" or "jsmith@4d.com"+Char(13)+"jdupont@4d.fr"
rutaLocal	Texto	→	- Documento Mac: "My Hard Drive:4DDB:SalesDB:Report" Win: "C:\MyDrive\4DDB\SalesDB\Report.txt" - Directorio Mac: "My Hard Drive:CoolStuff:" (Note el ":" final) Win: "C:\MyDrive\CoolStuff\"
rutaServidor	Text	→	- Documento "/usr/jsmith/reports/salesreport.txt" - Directorio "/usr/jsmith/reports/"(Note trailing "/")
tcp_ID	Entero largo	→	Referencia de una sesión TCP abierta
smtp_ID	Entero largo	→	Referencia de un nuevo mensaje
pop3_ID	Entero largo	→	Referencia de una sesión POP3 abierta
imap_ID	Entero largo	→	Referencia de una conexión IMAP abierta
ftp_ID	Entero largo	→	Referencia de una sesión FTP abierta
udp_ID	Entero largo	→	Referencia de una sesión UDP
Resultado	Entero	←	Código de error

hostName

El parámetro *nomServidor* es el nombre o la dirección IP del servidor local (HostName), por ejemplo "dns.4d.com" o "204.118.90.2". Los nombres de servidores se convierten por medio de un sistema de nombres de dominio. Los nombres de dominios por defecto (primarios) y secundarios normalmente se definen en el panel de control del driver TCP/IP instalado. Todo comando de Internet 4D necesita un *nomServidor* como parámetro aceptando su nombre ("www.4d.com") o su dirección IP ("204.118.90.2") . El formato "nombre" siempre es preferible, ya que protege a la aplicación de efectos nocivos debidos a los cambios de hardware en sitios remotos.

ip_Entero largo

Los nombres de los servidores pueden resolverse vía los métodos descritos anteriormente aplicados a una dirección IP. Las fórmulas matemáticas pueden entonces aplicarse a las direcciones IP para convertirlas en enteros largor únicos. Los comandos *NET_NameToAddr* y **NET_AddrToName** automatizan esta conversión. Este valor entero largo es designado como *ip_EnteroLargo* en esta documentación. Este valor sólo es utilizado en circunstancias especiales por los desarrolladores que establecen comunicación TCP directa. Algunos desarrolladores también prefieren almacenar el valor entero largo de un nombre de dominio para conservar el espacio en disco comparado con su cadena

equivalente. Sin embargo, por razones de compatibilidad con IPV6, 4D no aconseja a los desarrolladores utilizar esta funcionalidad.

mailAddress

El parámetro *direccionEmail* es una especificación completa de correo electrónico en el formato "nombre_usuario@nombre_dominio". En este documento, *direccionEmail* se refiere a una sola dirección de correo electrónico. Todo parámetro 4D Internet Commands que pueda tomar más de una dirección especificará *direccionEmail*. Si un parámetro tiene *direccionEmail* como su único tipo, puede tomar una y sólo una dirección de correo electrónico. El formato de *direccionEmail* debe ser una referencia completa que contenga tanto el nombre de usuario y como el nombre de dominio:

- "Felix Unger" <felix@pristine.com>
- oscar@slobs.com (Oscar Madison)

addressList

El parámetro *listaDirecciones* contiene una o más direcciones electrónica en el formato de *direccionEmail*, separadas por coma o un retorno de carro. La delimitación por retorno de carro es útil para ofrecer a los usuarios un área de texto que permita introducir o pegar varias direcciones. Los siguientes tres ejemplos generan un valor *$listaDirecciones* valido:

```
$AddressList:="jsmith@4d.com"
$AddressList:="jsmith@4d.com,scott@4d.com,marcel@4d.fr"
For($i;1;Size of array(aAddresses))
    $AddressList:=$AddressList+aAddresses{$i}+Char(13)
End for
```

rutaLocal

El parámetro *rutaLocal* indica la ubicación de un archivo o de un directorio en el ordenador del usuario (Mac o Windows).

En un Macintosh, los elementos al interior de las carpetas están separados por los caracteres "dos puntos" (:). Por ejemplo, el archivo "Mi informe" en la carpeta "Informes" en el disco duro "Mi disco duro" será como ruta de acceso "Mi disco duro:Informes:Mi informe". Una especificación de directorio en un Macintosh debe terminar con un carácter "dos puntos". Por ejemplo, si quiere poner un nuevo informe en la carpeta indicada anteriormente, debe pasar la cadena "Mi disco duro:Informes:". La decisión de hacer referencia a un nombre de archivo o de directorio está relacionada con el contexto del comando.

Bajo Windows, el principio utilizado es idéntico, a excepción de que una barra oblicua inversa "\" se utiliza en lugar de "dos puntos".

Nota: con el protocolo FTP, los nombres de los archivos manipulados por los comandos Internet de 4D tienen un tamaño limitado. Para mayor información consulte la sección (ver **Transferencia de archivos, Presentación**).

rutaServidor

La *rutaServidor* es la ubicación de un archivo o de un directorio en un ordenador que funciona bajo el sistema operativo Unix. En el entorno Unix, los directorios están separados por barras oblicuas ("/"). Por ejemplo, el archivo "informe.txt" en el directorio "informes" del directorio "4D" será designado por "/4D/informes/informes.txt". La ruta de acceso de un directorio debe terminar con un carácter "/". Note que una ruta de acceso completa comienza por una barra oblicua "/" que representa la raíz del volumen.

Nota: con el protocolo FTP, los comandos Internet de 4D trabajan con archivos cuyos nombres tienen un tamaño limitado. Para más información, consulte la sección **Transferencia de archivos,**

Presentación).

smtp_ID, pop3_ID, imap_ID, ftp_ID, tcp_ID

A lo largo de cada sección de 4D Internet Commands, se hace referencia a un número "ID" en la mayoría de los comandos. Cada conjunto de funciones de comunicación establecerán su propia "sesión", representada por un número entero largo "ID". Los comandos posteriores relacionados con la sesión abierta usarán este valor para dirigir sus efectos por el canal adecuado.

Los números "ID" obtenidos en cada sección (SMTP, POP3, IMAP, FTP, TCP, UDP), no se pueden pasar como valores a las diferentes secciones. Sin embargo, para mayor flexibilidad, 4D Internet Commands le permite pasar una referencia de conexión POP3, IMAP o FTP directamente a comandos TCP de bajo nivel y viceversa. Para obtener más información, consulte la sección **Rutinas de bajo nivel, Presentación**.

Referencia de sesión	Abierta por	Cerrada por
tcp_ID	TCP_Open or TCP_Listen	TCP_Close
smtp_ID	SMTP_New	SMTP_Clear
pop3_ID	POP3_Login	POP3_Logout or POP3_VerifyID
imap_ID	IMAP_Login	IMAP_Logout or IMAP_VerifyID
ftp_ID	FTP_Login	FTP_Logout or FTP_VerifyID
udp_ID	UDP_New	UDP_Delete

Resultado

Todos los comandos de Internet 4D (con excepción de **IT_ErrorText** y **IT_Version**) devuelven un valor entero como resultado de la función. Este entero contiene todo número de error que el comando deba transmitir de nuevo a la base de datos 4D. Si un comando se ejecuta correctamente, se devuelve un cero. De lo contrario, se devuelve un código de error. Para obtener más información sobre los códigos de error de 4D Internet Commands, consulte **Anexo C, Códigos de error de 4D Internet Commands**.

IC Downloaded Mail

- Descarga de correo, presentación
- MSG_Charset
- MSG_Delete
- MSG_Extract
- MSG_FindHeader
- MSG_GetBody
- MSG_GetHeaders
- MSG_GetMessage
- MSG_GetPrefs
- MSG_HasAttach
- MSG_MessageSize
- MSG_SetPrefs

Descarga de correo, presentación

Los comandos con prefijo "MSG_" permiten al usuario manipular los mensajes de correo que han sido guardados como archivos locales utilizando los comandos *POP3_Download* o *IMAP_Download* descritos en la sección anterior. 4D Internet Commands es totalmente compatible con MIME, puede extraer los archivos adjuntos. Para obtener más información sobre las normas MIME, consulte los documentos RFC1521, RFC1522 y RFC2045.

Una vez descargados los mensajes a archivos locales, los comandos de esta sección ofrecen una variedad de funciones para manipular los documentos. Estos comandos pueden obtener información sobre las partes del mensaje, por separar el encabezado del cuerpo del mensaje, detectar y extraer los archivos adjuntos, así como también eliminar los documentos existentes.

MSG_Charset

MSG_Charset (decodEncab ; conjCuerpos) -> resultado

Parámetro	Tipo		Descripción
decodEncab	Entero	→	-1 = Utilizar el parámetro actual, 0 = No hace nada, 1 = Convertir en el conjunto de caracteres Mac OS si ISO-8859-1 o ISO-2022-JP, decodificar los caracteres extendidos
conjCuerpos	Entero	→	-1 = Utilizar el parámetro actual, 0 = No hace nada, 1 = Convertir en el conjunto de caracteres Mac OS si ISO-8859-1 o ISO-2022-JP
resultado	Entero	↩	Código de error

Descripción

El comando *MSG_Charset* automatiza el tratamiento de los caracteres extendidos en los mensajes durante el proceso con los comandos MSG. Si este comando no se llama o los parámetros están en 0, 4D Internet Commands versión 6.8.1 o superior funcionará del mismo modo que la versión 6.5.x.

MSG_Charset permite, en primer lugar, definir si los encabezados con caracteres extendidos deben ser decodificados y en segundo lugar, si debe convertirse el conjunto de caracteres utilizado en el cuerpo del mensaje y en los encabezados.

Este comando es especialmente útil para el tratamiento de caracteres extendidos incluidos en los encabezados tales como el "Asunto" o direcciones de correo electrónico (por ejemplo, para decodificar una dirección como *"=?ISO-8859-1?Q?Test=E9?= <test@n.net >"*).

El parámetro *decodEncab* define el tratamiento a aplicar a los campos de encabezado durante la ejecución del comando *MSG_FindHeader*. El valor por defecto es 0.

- -1: Usar la configuración actual;
- 0: No hacer nada;
- 1: Los encabezados son decodificados si es necesario. Si el encabezado es decodificado y si el conjunto de caracteres especificado es ISO-8859-1 o ISO-2022-JP, los encabezados se convierten utilizando código ASCII Mac OS o Shift-JIS, respectivamente.

El parámetro *conjCuerpos* define el tratamiento a aplicar al cuerpo del mensaje durante la ejecución del comando *MSG_GetBody*. El valor por defecto es 0.

- -1: Usar la configuración actual;
- 0: No hacer nada;
- 1: Si el conjunto de caracteres especificado en el campo "Body-Content-Type" es ISO-8859-1 o ISO-2022-JP, el texto del cuerpo del mensaje se convierte utilizando ASCII Mac OS o Shift-JIS, respectivamente.

Nota de compatibilidad (versión 6.8.1): si el comando *MSG_Charset* no se utiliza y el comando *POP3_Charset* se ha utilizado, los comandos *MSG_FindHeader* y *MSG_GetBody* tendrán en cuenta los parámetros de *POP3_Charset*. Si se utiliza *MSG_Charset*, se ignoran los parámetros de *POP3_Charset*.

Ejemplo

Utilizando una versión 6.8.1 o superior de 4D Internet Commands:

```
$Err:=MSG_Charset(1;1)
$Err:=MSG_FindHeader($msgfile;"From";$from)
$Err:=MSG_FindHeader($msgfile;"To";$to)
$Err:=MSG_FindHeader($msgfile;"Cc";$cc)
$Err:=MSG_FindHeader($msgfile;"Subject";$subject)
$Err:=MSG_MessageSize($msgfile;$HdrSize;$BdySize;$msgSize)
$Err:=MSG_GetBody($msgfile;0;$BdySize;$BodyContent).
```

MSG_Delete

MSG_Delete (nomArchivo ; carpeta) -> resultado

Parámetro	Tipo		Descripción
nomArchivo	Texto	→	Nombre o ruta de acceso de archivo
carpeta	Entero	→	0 = Carpeta Messages, 1 = Carpeta Attachment
resultado	Entero	←	Código de error

Descripción

El comando *MSG_Delete* borra un archivo local.

El parámetro *nomArchivo* indica el nombre o la ruta de acceso completa del archivo a borrar. Si sólo se da un nombre de archivo, el parámetro *carpeta* se tiene en cuenta basado en:

- *carpeta* = 0: el archivo reside en la carpeta message definida por *POP3_SetPrefs* o *MSG_SetPrefs*.
- *carpeta* = 1: el archivo reside en la carpeta attachment definida por *POP3_SetPrefs* o *MSG_SetPrefs*
  En ambos casos, si *carpeta* no es definida por *POP3_SetPrefs* o *MSG_SetPrefs*, la carpeta utilizada será la que contiene el archivo de estructura de la base (con 4D monopuesto) o la carpeta de la aplicación 4D Client (con 4D Server).

Nota de compatibilidad (versión 6.8.1): si el comando *MSG_SetPrefs* no se utiliza, se utilizan los parámetros *msgFolder y attachFolder* del comando *POP3_SetPrefs*; si se utiliza *MSG_SetPrefs*, los parámetros del comando *POP3_SetPrefs* se ignoran.

Atención: este comando debe utilizarse con precaución porque borrará **TODO** archivo que se le pase.

MSG_Extract

MSG_Extract (nomArchivo ; decod ; rutaDocsAdj ; listAdj) -> resultado

Parámetro	Tipo		Descripción
nomArchivo	Texto	→	Nombre de archivo
decod	Entero	→	0 = No decodificar, 1 = Decodificar si es posible
rutaDocsAdj	Texto	→	Ruta de la carpeta (ruta por defecto en la carpeta attachment)
listAdj	Array cadena	←	Nombre de archivos adjuntos (sin ruta de acceso)
resultado	Entero	↩	Código de error

Descripción

El comando *MSG_Extract* extrae todos los archivos adjuntos y los pone en la carpeta de archivos adjuntos.

nomArchivo es el nombre o la ruta de acceso completa del archivo del cual extraer los archivos adjuntos. Si sólo se da un nombre de archivo, la ruta de acceso por defecto será la de la carpeta definida por *POP3_SetPrefs* o *MSG_SetPrefs* (ver la nota de compatibilidad). Si ninguna carpeta se ha especificado, la ruta por defecto será la de la carpeta que contiene la estructura de la base de datos (con 4D monousuario) o la de la carpeta 4D Client (con 4D Server).

El parámetro *decod* especifica si se debe intentar decodificar el archivo adjunto. Un valor de cero indica que no se debe intentar decodificar el archivo adjunto. Un valor de 1 tratará de decodificar el archivo si se ha codificado en una de las formas siguientes: BinHex, AppleSingle, AppleDouble, o Base64.

rutaDocsAdj indica la ruta de acceso de dónde guardar el archivo adjunto. Si pasa una cadena vacía, el archivo se guardará en la carpeta de archivos adjuntos especificada por *POP3_SetPrefs* o *MSG_SetPrefs* (ver la nota de compatibilidad). Si no se especifica carpeta, el archivo adjunto se guarda en la misma carpeta que la estructura de base de datos.

Nota de compatibilidad (versión 6.8.1): si el comando *MSG_SetPrefs* no se utiliza, se tienen en cuenta los parámetros *carpetaMsg y rutaDocsAdj* del comando *POP3_SetPrefs*; si se utiliza *MSG_SetPrefs*, los parámetros *carpetaMsg y rutaDocsAdj* del comando *POP3_SetPrefs* se ignoran.

listAdj es un array alfanumérico/texto que se devuelve con los nombres de los archivos de cada archivo adjunto. Sólo el nombre del documento será devuelto en cada elemento del array, sin la ruta de acceso.

MSG_FindHeader

MSG_FindHeader (nomArchivo ; etiqEncab ; valorEncab) -> resultado

Parámetro	Tipo		Descripción
nomArchivo	Texto	➡	Nombre de archivo
etiqEncab	Cadena	➡	Etiqueta del encabezado("De:", "Para:", "Asunto:", etc.)
valorEncab	Texto	⬅	Valor
resultado	Entero	↩	Código de error

Descripción

Dado el *nomArchivo* de un documento de mensaje recuperado en el disco por el comando *POP3_Download* o *IMAP_Download*, el comando *MSG_FindHeader* buscará la sección de encabezado de *etiqEncab* y devolver el valor asignado al campo en *valorEncab*.

nomArchivo es el nombre o ruta de acceso completa del archivo del cual extraer la información del encabezado. Si sólo se da un nombre de archivo, la ruta de acceso por defecto será la de la carpeta definida por *POP3_SetPrefs* o *MSG_SetPrefs* (ver nota de compatibilidad). Si ninguna carpeta ha sido especificada por *POP3_SetPrefs*, la ruta por defecto será la de la carpeta que contiene la estructura de la base de datos (con 4D monousuario) o la carpeta de 4D Client (con 4D Server).

Nota de compatibilidad (versión 6.8.1): si el comando *MSG_SetPrefs* no se utiliza, el parámetro *carpetaMsg* definido por el comando *POP3_SetPrefs* se tendrá en cuenta. Si el comando *MSG_SetPrefs* se utiliza, el parámetro definido por el comando *POP3_SetPrefs* se ignora.

etiqEncab es una cadena que contiene el nombre de la etiqueta de encabezado. etiqEncab puede hacer referencia a todo encabezado definido, especificado por el usuario o extendido, tal como "De:", "Para:", "X-MiEncabezdo", etc.

valorEncab es una variable de texto donde el comando devolverá el valor asignado al campo de encabezado especificado. Dado que el parámetro valorEncab puede incluir caracteres extendidos, puede automatizar su gestión con el comando *POP3_Charset* o *MSG_Charset*.

Nota de compatibilidad (versión 6.8.1): si el comando *MSG_Charset* no se utiliza, el parámetro *conjCuerpos* definido por el comando *POP3_Charset* se utilizará. Si se utiliza el comando *MSG_Charset*, se ignora el parámetro definido por el comando *POP3_Charset*.

MSG_GetBody

MSG_GetBody (nomArchivo ; offset ; longitud ; textoCuerpo) -> Resultado

Parámetro	Tipo		Descripción
nomArchivo	Texto	→	Nombre de archivo
offset	Entero largo	→	Inicio del offset en el cuerpo del mensaje (0= inicio del cuerpo)
longitud	Entero largo	→	Número de caracteres
textoCuerpo	Texto	←	Texto de cuerpo (elimina los retornos de línea si Prefs ON)
Resultado	Entero largo	↩	Código de error

Descripción

El comando *MSG_GetBody* devuelve sólo el texto del mensaje. No incluye el texto de los archivos adjuntos y borra todos los encabezados MIME.

nomArchivo es el nombre o ruta de acceso completa del archivo del cual extraer el cuerpo del mensaje. Si sólo pasa un nombre de archivo, la ruta de acceso por defecto será la de la carpeta definida por *POP3_SetPrefs* o *MSG_SetPrefs* (ver notas de compatibilidad). Si ninguna carpeta es especificada por *POP3_SetPrefs*, la ruta por defecto será la de la carpeta que contiene la estructura de la base de datos (con 4D monousuario) o la carpeta de 4D Client (con 4D Server).

El parámetro *offset* permite definir, en el cuerpo, la posición del carácter a partir del cual comenzar la recuperación.

El parámetro *longitud* es el número de caracteres a devolver.

El parámetro *textoCuerpo* recibe el texto del cuerpo del mensaje. Este parámetro puede incluir caracteres extendidos, puede automatizar su gestión con el comando *POP3_Charset* o *MSG_Charset* (ver notas de compatibilidad). Este parámetro tiene en cuenta el valor del parámetro *retornoLinea* definido por *POP3_SetPrefs* o *MSG_SetPrefs* (ver notas de compatibilidad).

Notas de compatibilidad (versión 6.8.1):

- Si el comando *MSG_SetPrefs* no se utiliza, se tendrán en cuenta los parámetros *carpetaMsg* y *retornoLinea* definidos por el comando *POP3_SetPrefs*. Si se utiliza *MSG_SetPrefs*, se ignoran los parámetros definidos por el comando *POP3_SetPrefs*.
- Si el comando *MSG_Charset* no se utiliza, se tiene en cuenta el parámetro *bodyCharset* definido por el comando *POP3_Charset*; si se utiliza el comando *MSG_Charset*, se ignora el parámetro *bodyCharset* definido por el comando *POP3_Charset*.

MSG_GetHeaders

MSG_GetHeaders (nomArchivo ; offset ; longitud ; textoEncab) -> resultado

Parámetro	Tipo		Descripción
nomArchivo	Texto	→	Nombre de archivo
offset	Entero largo	→	Inicio del offset en la sección de encabezados(0= inicio del encabezado)
longitud	Entero largo	→	Número de caracteres
textoEncab	Texto	←	Texto del encabezado (elimina los retornos de línea si Prefs están activadas)
resultado	Entero	↩	Código de error

Descripción

El comando *MSG_GetHeaders* devuelve el texto bruto de la sección de encabezado del mensaje. El encabezado de un mensaje POP3 se define como el texto del principio del mensaje hasta la primera ocurrencia de dos secuencias consecutivas de retorno de carro/retorno línea.

nomArchivo es el nombre o la ruta de acceso completa del archivo del cual extraer el encabezado. Si sólo se da un nombre de archivo, la ruta de acceso por defecto será la de la carpeta definida por *POP3_SetPrefs* o *MSG_SetPrefs* (ver la nota de compatibilidad). Si ninguna carpeta se ha especificado, la ruta por defecto será la de la carpeta que contiene la estructura de la base de datos (con 4D monousuario) o la de la carpeta 4D Client (con 4D Server).

El parámetro *offset* permite definir, en el cuerpo, la posición del carácter a partir del cual comenzar la recuperación.

El parámetro *longitud* es el número de caracteres a devolver. La longitud de la sección de encabezado se puede determinar con *MSG_MessageSize*.

El parámetro *textoEncab* recibe el texto del encabezado. Este parámetro tiene en cuenta el valor del parámetro retornoLinea definido por *POP3_SetPrefs* o *MSG_SetPrefs*.

Nota de compatibilidad (versión 6.8.1): si el comando *MSG_SetPrefs*, no se utiliza, el parámetro *carpetaMsg* definido por el comando *POP3_SetPrefs* se tendrá en cuenta; Si se utiliza el comando *MSG_SetPrefs*, se ignora el parámetro definido por el comando *POP3_SetPrefs*.

MSG_GetMessage

MSG_GetMessage (nomArchivo ; offset ; longitud ; textoBruto) -> resultado

Parámetro	Tipo		Descripción
nomArchivo	Texto	→	Nombre de archivo
offset	Entero largo	→	Inicio del offset en el archivo del mensaje (0= inicio del archivo)
longitud	Entero largo	→	Número de caracteres
textoBruto	Texto	←	Texto bruto (ignora las Prefs)
resultado	Entero	↩	Código de error

Descripción

El comando *MSG_GetMessage* devuelve el texto sin formato del mensaje independientemente de los datos adjuntos. No elimina los encabezados MIME.

nomArchivo es el nombre o la ruta de acceso completa del archivo del cual extraer el cuerpo del mensaje. Si sólo pasa un nombre de archivo, la ruta de acceso por defecto será la de la carpeta definida por *POP3_SetPrefs* o *MSG_SetPrefs* (ver Nota de compatibilidad). Si no se especifica ninguna carpeta, la ruta por defecto será a la carpeta que contiene la estructura de la base.

Nota de compatibilidad (versión 6.8.1): si no se utiliza el comando *MSG_SetPrefs*, se tiene en cuenta el parámetro *carpetaMsg* definida por el comando *POP3_SetPrefs*; si se utiliza *MSG_SetPrefs*, se ignora el parámetro definido por el comando *POP3_SetPrefs*.

El parámetro *offset* permite definir, en el mensaje fuente, la posición del carácter a partir del cual comenzar la recuperación.

El parámetro *longitud* es el número de carácteres a devolver.

El parámetro *textoBruto* recibe todo el texto del mensaje. Se ignoran los parámetros de las preferencias para la eliminación de los retornos de línea y los posibles documentos adjuntos en el cuerpo del mensaje no se eliminan.

MSG_GetPrefs

MSG_GetPrefs (retornoLinea ; carpetaMsg ; carpetaDocsAdj) -> resultado

Parámetro	Tipo		Descripción
retornoLinea	Entero	⬅	0 = No retirar los retornos de línea, 1 = Retirar los retornos de línea
carpetaMsg	Texto	⬅	Ruta de acceso a la carpeta de mensajes ("" = sin cambios)
carpetaDocsAdj	Texto	⬅	Ruta de acceso a la carpeta de los documentos adjuntos ("" sin modificación)
resultado	Entero	↩	Código de error

Descripción

El comando *MSG_GetPrefs* permite conocer las preferencias actuales para los comandos MSG.

Las preferencias se devuelven en las variables listadas en los parámetros.

El parámetro *retornoLinea* devuelve el parámetro actual de la opción de eliminación de los retornos de línea.

El parámetro *carpetaMsg* devuelve la ruta de acceso local de la carpeta en la cual se registran por defectos los mensajes recuperados.

El parámetro *carpetaDocsAdj* devuelve la ruta de acceso local de la carpeta en la cual se guardan por defecto los documentos adjuntos extraídos de los mensajes.

MSG_HasAttach

MSG_HasAttach (nomArchivo ; NumDocsAdj) -> resultado

Parámetro	Tipo		Descripción
nomArchivo	Texto	➡	Nombre de archivo
NumDocsAdj	Entero	⬅	Número de documentos adjuntos
resultado	Entero	↩	Código de error

Descripción

Si el archivo tiene documentos adjuntos, el comando *MSG_HasAttach* devuelve en el parámetro *numDocAdj* el número de documentos adjuntos al mensaje. Un documento adjunto es una pieza adjunta que no está en texto MIME. Si el mensaje no tiene documentos adjuntos, devuelve 0.

nomArchivo es el nombre del archivo o la ruta completa del archivo en el cual verificar la presencia de los documentos adjuntos. Si sólo pasa un nombre de archivo, la ruta de acceso por defecto será la del archivo definido por *POP3_SetPrefs* o *MSG_SetPrefs* (ver Nota de compatibilidad). Si no se ha especificado una carpeta, la ruta por defecto de la carpeta contiene la estructura de la base (con 4D monousuario) o de la carpeta de 4D Client (con 4D Server).

Nota de compatibilidad (versión 6.8.1): si el comando *MSG_SetPrefs* no se utiliza, se tiene en cuenta el parámetro *carpetaMsg* definido por el comando *POP3_SetPrefs*; si se utiliza el comando *MSG_SetPrefs*, se ignora el parámetro definido por *POP3_SetPrefs*.

El parámetro *numDocsAdj* devuelve el número de documentos adjuntos para *nomArchivo*.

MSG_MessageSize

MSG_MessageSize (nomArchivo ; tamEncabezado ; tamMsg ; tamMsg) -> resultado

Parámetro	Tipo		Descripción
nomArchivo	Texto	→	Nombre de archivo
tamEncabezado	Entero largo	←	Tamaño de la sección de encabezado (elimina los retornos de línea si Prefs ON)
tamMsg	Entero largo	←	Tamaño del cuerpo (ignora las Prefs)
tamMsg	Entero largo	←	Tamaño del mensaje o del archivo (ignora las Prefs)
resultado	Entero	↩	Código de error

Descripción

El comando ***MSG_MessageSize*** reenvía información sobre el tamaño de las diferentes partes del mensaje dado por *nomArchivo. nomArchivo* contiene el nombre y/o la ruta de acceso de un mensaje descargado localmente por el comando *POP3_Download*.

nomArchivo es el nombre o la ruta de acceso completa del archivo del cual extraer la información del mensaje. Si pasa sólo un nombre de archivo, la ruta de acceso por defecto será la de la carpeta definida por *POP3_SetPrefs* o *MSG_SetPrefs* (ver nota de compatibilidad). Si ninguna carpeta ha sido especificada por *POP3_SetPrefs*, la ruta por defecto será la de la carpeta que contiene el archivo de estructura de la base de datos (con 4D monopuesto) o de la carpeta de 4D Client (con 4D Server).

tamEncab devuelve el tamaño de la sección de encabezado.
tamCuerpo devuelve el tamaño del cuerpo del texto.
Estos dos parámetros tienen en cuenta el valor del parámetro *retornoLinea* definido por *POP3_SetPrefs* o *MSG_SetPrefs*.

Nota de compatibilidad (versión 6.8.1): si el comando *MSG_SetPrefs* no se utiliza, se tienen en cuenta los parámetros *carpetaMsg* y *retornoLinea* definidos por el comando *POP3_SetPrefs*. Si se utiliza el comando *MSG_SetPrefs*, se ignoran los parámetros definidos por el comando *POP3_SetPrefs*.

El parámetro *tamMsg* devuelve el tamaño global del mensaje.

MSG_SetPrefs

MSG_SetPrefs (retornoLinea ; carpetaMsg ; carpetaDocsAdj) -> resultado

Parámetro	Tipo		Descripción
retornoLinea	Entero	→	0 = No eliminar los retornos de línea, 1 = Retirar los retornos de línea, -1 = No cambiar
carpetaMsg	Texto	→	Ruta de acceso a la carpeta de mensajes (“” = no cambiar)
carpetaDocsAdj	Texto	→	Ruta de acceso a la carpeta de documentos adjuntos (“” = no cambiar)
resultado	Entero	←	Código de error

Descripción

El comando *MSG_SetPrefs* define las preferencias para todos los comandos MSG.

retornoLinea permite precisar cómo tratar los caracteres de retorno de línea en los mensajes descargados. La mayoría de mensajes de Internet combinan caracteres retorno de carro y retorno de línea para indicar el final de una línea. Las aplicaciones Macintosh requieren un retorno de carro como carácter de fin de línea. Esta opción permite a los usuarios retirar los caracteres retorno de línea del texto de los mensajes. Un valor de cero dejará el mensaje "tal cual". Un valor de uno retira los caracteres retorno de línea de los mensajes recuperados. Un valor de -1 dejará esta preferencia tal cual estaba definida. Por defecto, esta opción está en 1 y los retornos de línea se eliminan automáticamente en los mensajes.

carpetaMsg es un valor de texto que indica la ruta de acceso local de la carpeta en la cual los mensajes recuperados se guardan por defecto.

Nota de compatibilidad (versión 6.8.1): si el comando *MSG_SetPrefs* no se utiliza, los parámetros *retornoLinea* y *carpetaMsg* del comando *POP3_SetPrefs* se tendrán en cuenta si este comando se ha utilizado previamente. Si el comando *MSG_SetPrefs* se utiliza, los parámetros *POP3_SetPrefs* se ignoran.

carpetaDocsAdj es un valor de texto que contiene la ruta de acceso local de la carpeta en la que se guardan los archivos adjuntos cuando el comando *MSG_Extract* extrae los documentos adjuntos del cuerpo del mensaje.

Nota de compatibilidad (versión 6.8.1): este parámetro también se encuentra en *POP3_SetPrefs*; por lo tanto, puede modificarlo utilizando cualquiera de estos dos comandos.
Le recomendamos que utilice el comando *MSG_SetPrefs*. El parámetro del comando *POP3_SetPrefs*, se conserva por razones de compatibilidad, no se utilizará en futuras versiones del plug-in (este parámetro es opcional). Esta recomendación se aplica también al comando *POP3_GetPrefs*.

IC File Transfer

- Transferencia de archivos, Presentación
- FTP_Append
- FTP_ChangeDir
- FTP_Delete
- FTP_GetDirList
- FTP_GetFileInfo
- FTP_GetPassive
- FTP_GetType
- FTP_Login
- FTP_Logout
- FTP_MacBinary
- FTP_MakeDir
- FTP_PrintDir
- FTP_Receive
- FTP_RemoveDir
- FTP_Rename
- FTP_Send
- FTP_SetPassive
- FTP_SetType
- FTP_System
- FTP_VerifyID
- *FTP_Progress*

Transferencia de archivos, Presentación

El protocolo FTP (File Transfer Protocol) es el principal medio de transmisión de documentos y aplicaciones de un ordenador a otro. Los "sitios" FTP son ordenadores en todo el mundo que ejecutan un software FTP servidor. El protocolo FTP ofrece un medio de intercambio de archivos entre diferentes sistemas. Las aplicaciones clientes en diferentes plataformas pueden acceder a un servidor FTP para subir o descargar archivos de texto o binarios. Las rutinas de FTP dentro de 4D Internet Commands ofrecen a los desarrolladores las herramientas para crear clientes FTP dentro de sus bases de datos 4D.

Notas:

- Durante la especificación de las rutas de acceso en los comandos FTP, siempre se debe definir la ubicación de los archivos en los sitios FTP como un directorio Unix, incluso si el servidor FTP es un Macintosh ejecutando un sofware servidor FTP. Cualquiera que sea la plataforma, el software del servidor FTP convertirá internamente su ruta de acceso Unix al formato requerido para enviar sus documentos a los clientes conectados.
- Para una mayor flexibilidad, 4D Internet commands permite pasar directamente una referencia de conexión POP3, IMAP o FTP directamente a los comandos TCP de bajo nivel y viceversa. Para mayor información, consulte la sección **Rutinas de bajo nivel, Presentación**
- Los comandos FTP de 4D Internet Commands trabajan con archivos cuyos nombres tienen una longitud limitada. En 4D Internet Commands v16 R2 y siguientes, los nombres de los archivos están limitados a 1.024 caracteres, cualquiera que sea el sistema operativo(*).
  (*) En las versiones de macOS 32 bits, 4D Internet Commands utiliza las APIs de gestión de archivos declaradas obsoletas y no mantenidas por Apple, Inc. En este entorno, dependiendo de la versión macOS, los nombres de los archifos pueden tener más limitaciones relativas a la longitud y a los caracteres soportados. Recomendamos actualizar a 4D Internet Commands 64 bits tan pronto como sea posible.

FTP_Append

FTP_Append (ftp_ID ; rutaLocal ; rutaServidor ; Progreso) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	→	Referencia de una conexión FTP
rutaLocal	Texto	→	Ruta de acceso local del documento a enviar
rutaServidor	Texto	→	Ruta de acceso local del documento a enviar
Progreso	Entero	→	*** Parámetro obsoleto (ignorado) ***
resultado	Entero	↩	Código de error

Descripción

El comando *FTP_Append* realiza la misma acción que *FTP_Send* con la diferencia de que añade los datos enviados al final del archivo existente identificado por *rutaServidor*. La principal función de este comando es añadir los datos al final de los archivos texto preexistentes.

FTP_ChangeDir

FTP_ChangeDir (ftp_ID ; rutaServidor) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	➡	Referencia de una conexión FTP
rutaServidor	Texto	➡	Ruta de acceso a un directorio Unix en el servidor FTP
resultado	Entero	⮌	Código de error

Descripción

El comando *FTP_ChangeDir* permite designar el directorio de trabajo actual (Current Working Directory o CWD) en el servidor FTP.

Nota: también es posible modificar el directorio de trabajo actual con ayuda de los comandos *FTP_GetDirList* y *FTP_GetFileInfo*. Sin embargo, el comando *FTP_ChangeDir* es más rápido y necesita menos parámetros.

ftp_ID es el identificador de la sesión FTP establecida con *FTP_Login*.

El parámetro *rutaServidor* contiene una ruta de acceso al formato Unix referenciando un directorio FTP existente y accesible. Si el directorio FTP especificado es inválido (inexistente o con derechos de acceso insuficientes), *FTP_ChangeDir* devuelve un error y no modifica el repertorio de trabajo actual.

Ejemplo

El siguiente ejemplo designa la raíz del servidor FTP como directorio de trabajo actual:

```
$err:=FTP_ChangeDir(ftp_ID;"/")
```

FTP_Delete

FTP_Delete (ftp_ID ; rutaServidor) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	➡	Referencia de una conexión FTP
rutaServidor	Texto	➡	Ruta de acceso del documento en el servidor FTP
resultado	Entero	⮌	Código de error

Descripción

Dada una ruta de acceso a un archivo en el formato de un rutaServidor, el comando *FTP_Delete* borrará el archivo especificado del servidor FTP remoto. Se devuelve un error si no tiene los privilegios de acceso suficientes para realizar esta acción.

ftp_ID es la referencia entero largo de la sesión FTP establecida con *FTP_Login*.

rutaServidor designa la ruta de acceso al formato Unix del documento a borrar. Puede pasar en este parámetro una ruta de acceso completa o un simple nombre de archivo. Si utiliza la forma abreviada, el archivo especificado debe estar en el directorio de trabajo actual (CWD).

Nota: puede cambiar el directorio de trabajo actual (CWD) con el comando *FTP_ChangeDir*. También puede conocer en cualquier momento el directorio de trabajo actual con la ayuda del comando *FTP_PrintDir*.

FTP_GetDirList

FTP_GetDirList (ftp_ID ; rutaServidor ; nombres ; tams ; tipos ; fechasMod {; horasMod}) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	➡	Referencia de una conexión FTP
rutaServidor	Texto	➡	Ruta de acceso a un directorio Unix en el servidor FTP
		⬅	Directorio de trabajo actual (CWD)
nombres	Array cadena	⬅	Lista de nombres
tams	Array entero largo	⬅	Lista de tamaños
tipos	Array entero	⬅	Lista de tipos 0 = archivo normal, 1 = directorio, 2 = archivo especial de tipo bloque, 3 = archivo especial de tipo carácter, 4 = enlace simbólico, 5 = archivo especial FIFO, 6 = puerto de acceso de la familia de direcciones AF_UNIX
fechasMod	Array fecha	⬅	Lista de las fechas de modificación
horasMod	Array entero largo	⬅	Lista de horas de modificación
resultado	Entero	↩	Código de error

Descripción

El comando *FTP_GetDirList* devuelve la lista de los objetos en un *directorio* de la sesión FTP identificada por *ftp_ID*. La información sobre los nombres, tamaños, tipos, fechas y horas de modificación de los elementos del directorio se devuelve en arrays. Una conexión al sitio FTP debe haber sido abierta por *FTP_Login* y seguir siendo válida (*FTP_VerifyID*). El comando *FTP_GetDirList* reemplaza al directorio de trabajo actual (CWD) por el definido en el parámetro *rutaServidor*.

ftp_ID es la referencia entero largo de la sesión FTP establecida con *FTP_Login*.

directorio es un valor de texto en el formato de una rutaServidor que hace referencia a un directorio FTP. Una variable o campo 4D debe pasarse en este parámetro ya que el directorio de trabajo actual resultante será devuelto después de la ejecución del comando. Normalmente, el valor devuelto a este parámetro será el mismo que el valor que se pasa. Sin embargo, pueden haber casos (como las restricciones de acceso) donde el cambio de directorio no se realiza correctamente. En este caso, el parámetro directorio tendrá la rutaServidor al directorio actual de la sesión.

Si pasa una cadena vacía en este parámetro, los arrays se llenan con las listas de los archivos del directorio actual y la ruta de acceso del directorio actual del servidor (CWD) se devuelve en el parámetro *rutaservidor*.

nombres es un array de tipo alfanumérico o texto que recibe el nombre de cada objeto en el *directorio* especificado.

tams es un array de tipo entero largo que recibe el tamaño de los objetos del *directorio.*

tipos es un array de tipo entero que recibe los valores de tipo de cada objeto del directorio. Estos son los valores posibles y los tipos correspondientes:

Tipo	Archivo
0	archivo ordinario
1	directorio
2	archivo especial de tipo bloque
3	archivo especial de tipo carácter
4	enlace simbólico (alias de los archivos o carpetas)
5	archivo especial FIFO
6	puerto de acceso de la familia AF_UNIX

Nota: en el caso de un enlace simbólico (tipo = 4), el servidor FTP devuelve una ruta particular (Nombre de alias + símbolo + ruta de acceso al archivo o carpeta fuente). Si intenta utilizar esta ruta para acceder al archivo o carpeta fuente, se devuelve un error. Debe extraer la ruta de acceso del archivo o de la carpeta de origen de la cadena devuelta por *FTP_GetDirList* que comienza justo después

del carácter simbólico. De lo contrario, los comandos tales como *FTP_GetFileInfo* devolverán el error -10085 ya que el archivo o carpeta no se encuentra.

fechasMod es una array de tipo fecha que recibe la fecha de la última modificación de cada objeto del *directorio*.

horasMod es un array de tipo entero largo que recibe la hora de la última modificación de cada objeto en el *directorio*.

Recordatorio: en 4D, el tipo de array entero largo se utiliza para manipular los datos de tipo hora (cada elemento del array representa un número de segundos). Utilice el comando **Time string** para convertir estos valores al formato **HH:MM:SS**.

FTP_GetFileInfo

FTP_GetFileInfo (ftp_ID ; rutaServidor ; tamaño ; fechaMod ; horasMod) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	→	Referencia de una conexión FTP
rutaServidor	Texto	→	Ruta de acceso del documento en el servidor FTP
tamaño	Entero largo	←	Tamaño del documento
fechaMod	Fecha	←	Fecha de modificación
horasMod	Hora	←	Hora de modificación
resultado	Entero	↩	Código de error

Descripción

El comando *FTP_GetFileInfo* devuelve la información sobre la última modificación del archivo designado por *rutaServidor.*

ftp_ID es el identificador de la sesión FTP establecida con *FTP_Login*.

rutaServidor contiene la ruta de acceso al documento del cual quiere obtener información.

Nota: el comando *FTP_GetFileInfo* puede modificar el directorio de trabajo actual (CWD) si *rutaServidor* es una ruta de acceso completa que indica un directorio diferente del directorio de trabajo actual. En este caso, el directorio definido por el parámetro *rutaServidor* se convierte en el directorio de trabajo actual.

tamaño devuelve el tamaño del archivo identificado por *rutaServidor*.

fechasMod y *horasMod* devuelven la fecha y la hora de la última modificación del archivo.

FTP_GetPassive

FTP_GetPassive (ftp_ID ; modoPasivo) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	→	Referencia de una conexión FTP
modoPasivo	Entero	←	Modo de transferencia de datos actual 0 = modo activo, 1 = modo pasivo
resultado	Entero	↩	Código de error

Descripción

El comando *FTP_GetPassive* devuelve el modo de transferencia en el canal de transferencia de datos. Para mayor información sobre los modos de transferencia FTP, consulte la descripción del comando *FTP_SetPassive*.

ftp_ID es el identificador de la sesión de FTP establecida con *FTP_Login*.

modoPasivo devuelve el modo de transferencia actual en el canal de transferencia de datos:

- si *modoPasivo* vale 0, el servidor FTP funciona en modo activo.
- si *modoPasivo* vale 1, el servidor FTP funciona en modo pasivo (modo por defecto).

FTP_GetType

FTP_GetType (ftp_ID ; ftp_Mode) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	→	Referencia de una conexión FTP
ftp_Mode	Cadena	←	"A" = Ascii; "I" = Image; "L 8" = Logical 8-bit
resultado	Entero	↩	Código de error

Descripción

El comando *FTP_GetType* devuelve información sobre el modo de transferencia FTP actual. El modo de transferencia puede fijarse utilizando el comando *FTP_SetType*.

ftp_ID es el identificador de la sesión FTP establecida con *FTP_Login*.

El parámetro *ftp_Mode* devuelve un código que indica el modo de transferencia FTP actual.

FTP_Login

FTP_Login (nomServidor ; nomUsuario ; contraseña ; ftp_ID ; textoBienvenida) -> resultado

Parámetro	Tipo		Descripción
nomServidor	Cadena	➡	Nombre o dirección IP del servidor FTP
nomUsuario	Cadena	➡	Nombre del usuario
contraseña	Cadena	➡	Contraseña
ftp_ID	Entero largo	⬅	Referencia de esta nueva sesión FTP
textoBienvenida	Texto	⬅	Texto de bienvenida FTP
resultado	Entero	↩	Código de error

Descripción

El comando *FTP_Login* establece una conexión con el servidor FTP en *nomServidor* y se conecta al sistema utilizando el nombre de usuario y contraseña suministrados *nomUsuario* y *contraseña*.

nomServidor es el nombre o la dirección IP del servidor remoto.

nomUsuario es el nombre de una cuenta de usuario reconocida por el servidor FTP. Muchos servidores FTP soportan el acceso de invitados con un nombre de usuario "anónimo". Si se conecta de forma anónima, se acostumbra dar su dirección de correo electrónico como contraseña.

contraseña es la contraseña para el usuario en el servidor FTP.

ftp_ID es el valor entero largo que identifica la sesión abierta. Este valor será utilizado por los siguientes comandos FTP. Este parámetro debe pasarse como una variable o campo 4D con el fin de aceptar los resultados devueltos.

textoBienvenida es un parámetro opcional que contiene el texto devuelto cuando el usuario se conecta al sistema. Muchos sitios FTP tienen un mensaje de bienvenida que aparece al momento de inicio de sesión. Si se especifica, este parámetro debe pasarse como una variable o campo 4D para aceptar los resultados devueltos.

Ejemplo

```
$OK:=False
Case of
   :(FTP_Login("ftp.4d.com";"anonymous";"dbody@aol.com";vFTP_ID;vFTP_Msg)#0)
   :(FTP_Send(vFTP_ID;"Mi disco duro:Documents ƒ:Informe de ventas de julio";"/pub/reports";1)#0)
   :(FTP_Logout(vFTP_ID)#0)
   Else
      $OK:=True  `Todos los comandos son ejecutados sin error
End case
```

Nota: para mayor información sobre este uso particular de la estructura **Case of**, consulte **Anexo A: Consejos de programación**.

FTP_Logout

FTP_Logout (ftp_ID) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	➡	Referencia de una conexión FTP
		⬅	0 = sesión cerrada correctamente
resultado	Entero	↩	Código de error

Descripción

Dada una referencia a una sesión FTP abierta, el comando *FTP_Logout* desconecta al usuario del servidor FTP y libera la memoria utilizada por la sesión. Este comando devolverá el valor 0 (cero) en el parámetro *ftp_ID* si el cierre de la sesión es exitoso.

ftp_ID es el identificador de la sesión FTP establecida con *FTP_Login*.

Ejemplo

```
If(FTP_Login("ftp.4d.com";"anonymous";vEmailID;vFTP_ID;vFTP_Msg)=1)
    $error:=FTP_Send(vFTP_ID;"Mi disco duro:Documents:Informe de ventas";"/pub/reports";1)
    $error:=FTP_Logout(vFTP_ID)
End if
```

FTP_MacBinary

FTP_MacBinary (ftp_ID ; modoMacBinary) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	➡	Referencia de una conexión FTP
modoMacBinary	Entero	➡	-1 = Obtener el parámetro actual, 1 = Activar, 0 = Desactivar
		⬅	Parámetro actual (si se pasa -1)
resultado	Entero	↩	Código de error

Descripción

El comando *FTP_MacBinary* activa/desactiva el modo MacBinary para las transferencias FTP utilizando *FTP_Send* y **FTP_Receive** en la sesión FTP actual identificada por *ftp_ID*.

El protocolo MacBinary es utilizado con frecuencia por clientes y servidores FTP Macintosh para facilitar la transferencia de datos o de archivos binarios que contienen a la vez partes de datos (data forks) y parte de recursos (resource fork).

Nota para usuarios Windows: es posible usar el protocolo MacBinary para las transferencias FTP en un entorno Windows sin embargo cabe señalar que a menudo no tiene sentido decodificar un archivo MacBinary en un ordenador PC. Los ordenadores con procesadores Intel no pueden almacenar archivos con data forks y resource forks. Como este formato es ajeno a la plataforma Windows, los archivos Mac OS que contienen una parte de recursos (resource fork) corren el riesgo de dañarse si se guardan en un formato no codificado.

ftp_ID es la referencia entero largo de la sesión FTP establecida con *FTP_Login*.

El parámetro *modoMacBinary* indica si se debe activar o no el modo de transferencia MacBinary. Este valor debe pasarse como una variable para que el comando pueda devolver el estado de las transferencias MacBinary después del intento de cambio. 1 activa el modo de transferencia MacBinary y cero lo desactiva. -1 devuelve en *modoMacBinary* el parámetro actual del modo de transferencia MacBinary (1 ó 0).

Atención: no todos los servidores FTP soportan el protocolo MacBinary, en este caso el error 10053 se devuelve en cada llamada al comando *FTP_MacBinary*, cualquiera que sea el valor del parámetro *modoMacBinary*. Los comportamientos descritos previamente no aplican más.

Ejemplo

Este ejemplo activa el protocolo MacBinary antes de la recepción de un archivo FTP. Si el archivo se recibe correctamente con el MacBinary activado, se decodifica en su formato original y se borra el documento MacBinary.

```
vUseMacBin:=-1
$error:=FTP_MacBinary(vFTP_ID;vUseMacBin)
If($error=10053)
   MacBinaryIsSupported:=False  `El servidor ftp no soporta el protocolo MacBinary
Else
   MacBinaryIsSupported:=True
End if


vLocalFile:=""
If(MacBinaryIsSupported)
   vUseMacBin:=1
   $error:=FTP_MacBinary(vFTP_ID;vUseMacBin)  `Activación de MacBinary para la descarga
End if
$error:=FTP_Receive(vFTP_ID;"MyApplication";vLocalFile;cbShowTherm)
If($error=0) & (vUseMacBin=1)  `Si recibe OK y el archivo está en formato MacBinary
   vDecodePath:=""
```

```
    If(IT_Decode(vLocalFile;vDecodePath;8)=0)  `Decodificación MacBinary
        DELETE DOCUMENT(vLocalFile)  `Si se decodifica la fuente con éxito, se borra el archivo fuente.
    End if
End if
```

FTP_MakeDir

FTP_MakeDir (ftp_ID ; rutaServidor) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	➡	Referencia de una conexión FTP
rutaServidor	Texto	➡	Ruta de acceso a un directorio Unix en el servidor FTP
resultado	Entero	⮌	Código de error

Descripción

El comando **FTP_MakeDir** crea un nuevo directorio definido por el parámetro *rutaservidor*. Se devuelve un error si no tiene los derechos necesarios para efectuar esta operación.

ftp_ID es la referencia (entero largo) de la sesión FTP establecida con *FTP_Login*.

rutaServidor designa la ruta de acceso en formato Unix del directorio FTP a crear. Este parámetro puede contener una ruta de acceso completa o un simple nombre de carpeta. Si se utiliza la forma abreviada, el directorio se crea en el directorio de trabajo actual CWD. El nombre del directorio *rutaServidor* **no** debe tener espacios vacíos.

Nota: puede modificar el directorio de trabajo actual (CWD) utilizando el comando *FTP_ChangeDir*. También puede conocer en cualquier momento el directorio de trabajo actual con ayuda del comando *FTP_PrintDir*.

FTP_PrintDir

FTP_PrintDir (ftp_ID ; rutaServidor) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	➡	Referencia de una conexión FTP
rutaServidor	Texto	⬅	Ruta de acceso Unix a un directorio del servidor FTP
resultado	Entero	↩	Código de error

Descripción

El comando *FTP_PrintDir* devuelve el directorio de trabajo actual (CWD) en el servidor FTP.

Nota: esta información también puede obtenerse con la ayuda de *FTP_GetDirList* pasando una cadena vacía en el parámetro *rutaServidor*. Sin embargo, el comando *FTP_PrintDir* es más rápido y necesita menos parámetros.

ftp_ID es el identificador de la sesión FTP establecida con *FTP_Login*.

El parámetro *rutaServidor* devuelve la ruta del directorio de trabajo actual (CWD).

Ejemplo

Este ejemplo devuelve el directorio de trabajo actual en la variable $Cwd.

```
$err:=FTP_PrintDir(ftp_ID;$Cwd)
```

FTP_Receive

FTP_Receive (ftp_ID ; rutaServidor ; rutaLocal ; Progreso) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	➡	Referencia de una conexión FTP
rutaServidor	Texto	➡	Ruta de acceso en el servidor FTP del documento a recibir
rutaLocal	Texto	➡	Ruta de acceso local de destino del documento
		⬅	Ruta de acceso del documento resultante (si se pasa "")
Progreso	Entero	➡	1 = Mostrar progreso, 0 = Ocultar progreso
resultado	Entero	↩	Código de error

Descripción

El comando *FTP_Receive* recibe por FTP un archivo de la ruta de acceso referenciada por *rutaServidor*. *FTP_Receive* devuelve el error -48 si el archivo ya existe en el directorio de destino.

ftp_ID es la referencia entero largo de la sesión FTP establecida con *FTP_login*.

servidorLocal es un valor texto que especifica la ruta de acceso Unix completa del documento a recibir. Si *rutaServidor* no es una ruta de acceso completa a un documento, el comando devolverá un error. Al igual que con todas las rutas de acceso a los documentos Unix, la ruta debe utilizar barras oblicuas como separadores ("/"). Para obtener más información, consulte la sección **Glosario y terminología**.

El parámetro *rutaLocal* especifica la ruta de acceso del documento a crear localmente. Si *rutaLocal* es una cadena vacía, se muestra la caja de diálogo estándar Guardar archivo y el nombre y la ruta de acceso del archivo resultante se devuelve en la variable *rutaLocal*. Si *rutaLocal* contiene sólo un nombre de archivo, el archivo se guardará en la misma carpeta que la estructura de la base de datos (con 4D monousuario) o en la carpeta 4D Client (con 4D Server). Al igual que con todas las rutas de acceso a documentos locales, la ruta debe estar separada por el delimitador correspondiente a la plataforma utilizada. Para obtener más información, consulte la sección **Glosario y terminología**.

 indica si el termómetro de progreso se debe mostrar o no. Pase 1 para mostrar termómetro de progreso ó 0 para ocultarlo.

Compatibilidad: el parámetro *progreso* es obsoleto y se ignora si se pasa.

Ejemplo

```
vUseMacBin:=-1
$error:=FTP_MacBinary(vFTP_ID;vUseMacBin)
If($error=10053)
   MacBinaryIsSupported:=False  `El servidor Ftp no soporta el protocolo MacBinary
Else
   MacBinaryIsSupported:=True
End if

vLocalFile:=""
If(MacBinaryIsSupported)
   vUseMacBin:=1
   $error:=FTP_MacBinary(vFTP_ID;vUseMacBin)  `Activa MacBinary para la descarga
   $error:=FTP_Receive(vFTP_ID;"CGMiniViewer.hqx";vLocalFile)
   If($error=0) & (vUseMacBin=1)
      vDecodePath:=""
      If(IT_Decode(vLocalFile;vDecodePath;8)=0)  `Decodificar MacBinary
         DELETE DOCUMENT(vLocalFile)  `Si la decodificación es exitosa, borrar el archivo fuente.
      End if
   End if
End if
```

FTP_RemoveDir

FTP_RemoveDir (ftp_ID ; rutaServidor) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	➡	Referencia de una conexión FTP
rutaServidor	Texto	➡	Ruta de acceso a un directorio Unix en el servidor FTP
resultado	Entero	⮌	Código de error

Descripción

El comando *FTP_RemoveDir* elimina el directorio designado por el parámetro rutaServidor. Se devuelve un error si no tiene los privilegios de acceso suficientes para realizar esta operación. Además, si intenta eliminar un directorio que contiene elementos se presentará en un error de seguridad.

ftp_ID es la referencia entero largo de la sesión FTP establecida con *FTP_Login*.

rutaServidor es un valor texto en el formato de una rutaServidor que hace referencia a un directorio FTP existente. Este parámetro puede contener una ruta de acceso completa o un simple nombre de carpeta. Si se utiliza la forma abreviada entonces el directorio especificado debe estar en el directorio de trabajo actual (CWD).

Nota: puede cambiar el directorio de trabajo actual (CWD) con el comando *FTP_ChangeDir*. También puede conocer en cualquier momento el directorio de trabajo actual utilizando el comando **FTP_PrintDir**.

FTP_Rename

FTP_Rename (ftp_ID ; rutaServidor ; nuevoNom) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	➡	Referencia de una conexión FTP
rutaServidor	Texto	➡	Ruta de acceso del documento en el servidor FTP
nuevoNom	Texto	➡	Nuevo nombre del documento
resultado	Entero	⮌	Código de error

Descripción

El comando *FTP_Rename* cambia el nombre del archivo especificado por rutaServidor en el servidor FTP remoto. Se genera un error si usted no tiene los derechos de acceso suficientes para realizar esta acción.

ftp_ID es la referencia entero largo de la sesión FTP establecida con *FTP_Login*.

rutaServidor designa la ruta de acceso al formato Unix del documento a renombrar. Este parámetro puede contener una ruta de acceso completa o un simple nombre de archivo. Si se utiliza la forma abreviada, el archivo especificado debe estar en el directorio de trabajo actual (CWD).

Nota: puede cambiar el directorio de trabajo actual (CWD) con el comando *FTP_ChangeDir*. También puede conocer en cualquier momento el directorio de trabajo actual utilizando el comando *FTP_PrintDir*.

El parámetro *nuevoNombre* contiene el nombre con el cual quiere renombrar el documento remoto.

FTP_Send

FTP_Send (ftp_ID ; rutaLocal ; rutaServidor ; Progreso) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	➡	Referencia de una conexión FTP
rutaLocal	Texto	➡	Ruta de acceso local del documento a enviar
rutaServidor	Texto	➡	Ruta de acceso del documento en el servidor FTP
Progreso	Entero	➡	1 = Mostrar progreso, 0 = Ocultar progreso
resultado	Entero	⮌	Código de error

Descripción

El comando *FTP_Send* envía el documento designado por *rutaLocal* a la ubicación designada por *rutaServidor*. Si se produce un error de estado de archivo FTP, *FTP_Send* lo devuelve inmediatamente.

ftp_ID es la referencia entero largo de la sesión FTP establecida con *FTP_Login*.

rutaLocal es la ruta del documento a enviar. Si pasa una cadena vacía, aparece la caja de diálogo estándar de apertura de archivos. Si *rutaLocal* es un nombre de archivo simple (sin ruta de acceso), el comando busca este archivo en la carpeta que contiene el archivo de estructura de la base de datos (con 4D monopuesto) o en la carpeta de 4D Client (con 4D Server). Al igual que con todas las rutas de acceso a documentos locales, los directorios deben estar separados por un delimitador apropiado para la plataforma. Para obtener más información, consulte la sección **Glosario y terminología**.

Nota: los comandos FTP trabajan con documentos cuyos nombres tienen una longitud limitada. Para obtener más información, consulte la sección **Transferencia de archivos, Presentación**.

rutaServidor designa la ruta de acceso al formato Unix del documento a crear. *rutaServidor* indica el nuevo nombre del archivo una vez recibido por el servidor FTP. Si *rutaLocal* es una cadena vacía que permite al usuario elegir un archivo del disco, *rutaServidor* también puede ser una cadena vacía, en cuyo caso se utilizará el nombre del archivo seleccionado.

rutaServidor puede ser una ruta de acceso Unix completa o simplemente un nombre de archivo:

- Si pasa una ruta de acceso completa, el archivo especificado se ubicará en el directorio indicado por *rutaServidor*.
- Si sólo pasa un nombre de archivo, o cadenas vacías en la selección del archivo, el archivo será enviado al directorio de trabajo actual [CWD].

Si el archivo o la ruta no pueden interpretarse correctamente, el comando devolverá un error. Si el usuario no tiene los privilegios suficientes para enviar un archivo a ese directorio, se devuelve un error. Al igual que con todas las rutas de acceso a los documentos Unix, la ruta debe utilizar barras oblicuas como separadores ("/"). Para obtener más información, consulte la sección **Glosario y terminología**.

Nota: el servidor FTP también puede imponer limitaciones específicas relativas a la longitud o caracteres para los nombres de archivo.

Compatibilidad: el parámetro *progreso* es obsoleto y se ignora si se pasa.

Ejemplo 1

```
$OK:=False
Case of
   :(FTP_Login("ftp.4d.com";"anonymous";vEmailID;vFTP_ID;vFTP_Msg)#0)
   :(FTP_Send(vFTP_ID;"My Hard Drive:Documents:July Sales Report";"/pub/reports/July_sales";1)#0)
   :(FTP_Logout(vFTP_ID)#0)
   Else
      $OK:=True  `todos los comandos son ejecutados sin error
End case
```

Nota: para mayor información sobre este uso particular de la estructura **Case of**, consulte el **Anexo A: Consejos de programaciónIs Windows**.

Ejemplo 2

```
$error:=FTP_Send(vFTP_ID;"";"")
```

FTP_SetPassive

FTP_SetPassive (ftp_ID ; modoPasivo) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	➡	Referencia a una conexión FTP
modoPasivo	Entero	➡	0 = modo activo, 1 = modo pasivo (modo por defecto)
resultado	Entero	⮌	Código de error

Descripción

El comando *FTP_SetPassive* permite definir las modalidades de transferencia de datos entre un servidor FTP y un cliente de FTP, durante el uso de comandos tales como *FTP_GetDirList*, *FTP_Send*, *FTP_Append* o *FTP_Receive*. El modo de transferencia especificado es utilizado por estos comandos una vez se ejecute el comando *FTP_SetPassive*.

Los intercambios entre un servidor FTP y un cliente FTP tienen dos partes: intercambios en el canal de control (puerto 21 por defecto) e intercambios en el canal de transferencia de datos (puerto 20 por defecto). Por lo general, los servidores FTP se definen como "activos", ya que se encargan de la gestión de la transferencia en el canal de datos.
Por razones históricas, 4D Internet Commands pide a los servidores FTP trabajar en modo pasivo. Esto significa que antes de cada intercambio en el canal de transferencia de datos, se envía el comando FTP "PASV".
Sin embargo, algunos servidores FTP no soportan el modo pasivo, así como tampoco los sistemas de protección (firewalls). En estos casos, el comando FTP_SetPassive permite definir el modo activo para la transferencia de los datos.

Nota: se recomienda consultar con el administrador de red el modo de transferencia utilizado para los intercambios FTP.

ftp_ID es la referencia entero largo de la sesión FTP establecida con *FTP_Login*.

El parámetro *modoPasivo* especifica el modo de transferencia en el canal de transferencia de datos:

- Un valor de 0, pide al servidor FTP trabajar en modo activo
- Un valor de 1, pide al servidor FTP trabajar en modo pasivo (valor por defecto).

FTP_SetType

FTP_SetType (ftp_ID ; ftp_Mode) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	→	Referencia de una conexión FTP
ftp_Mode	Cadena	→	"A" = Ascii; "I" = [Default] Image; "L 8" = Logical 8-bit
resultado	Entero	↩	Código de error

Descripción

El comando *FTP_SetType* se utiliza para modificar el modo de transferencia FTP utilizado durante las operaciones enviar y recibir. Normalmente no es necesario cambiar este parámetro. Sin embargo, debido a las diferencias entre las distintas plataformas y versiones de FTP, puede ser necesario cambiar el modo para ciertos tipos de transferencias FTP. En particular, algunas transferencias de documentos en texto pueden necesitar el modo ASCII para transferir correctamente el archivo texto.

ftp_ID es la referencia entero largo de la sesión FTP establecida con *FTP_Login*.

ftp_Mode debe contener un código que indica el modo de transferencia a utilizar. Por defecto, se utiliza el modo Image ("I").

FTP_System

FTP_System (ftp_ID ; infoSistema) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo	→	Referencia de una conexión FTP
infoSistema	Cadena	←	Información del sistema
resultado	Entero	↩	Código de error

Descripción

El comando*FTP_System* devuelve en el parámetro *infoSistema* información que describe el software del servidor FTP.

ftp_ID es el identificador de la sesión FTP establecida con *FTP_Login*.

FTP_VerifyID

FTP_VerifyID (ftp_ID) -> resultado

Parámetro	Tipo		Descripción
ftp_ID	Entero largo		Referencia de una conexión FTP
			0 = Conexión cerrada
resultado	Entero		Código de error

Descripción

Un servidor FTP desconecta automáticamente las cuentas que no muestran actividad en un período de tiempo determinado por el administrador. Cada comando que interactúa con el servidor FTP provoca un reinicio de su contador de inactividad. El comando *FTP_VerifyID* provoca igualmente el reinicio a cero del contador de inactividad de la FTP especificada. Esto permite al usuario mantener una sesión activa si existe la posibilidad de que la sesión sobrepase el tiempo de inactividad definido en el servidor.

Durante su ejecución, el comando *FTP_VerifyID* verifica que la conexión no se haya cerrado. Si la sesión aún está abierta, el comando le indica al servidor FTP que reinicie el contador en cero para la sesión. Si la conexión ya se ha cerrado, *FTP_VerifyID* devuelve el error correspondiente, libera la memoria usada por la sesión FTP y vuelve a cero nuevamente el parámetro *ftp_ID*.

ftp_ID es la referencia de la sesión FTP establecida con *FTP_Login*.

FTP_Progress

FTP_Progress (izq ; superior ; tituloVentana ; textoTermo ; cancelar) -> resultado

Parámetro	Tipo		Descripción
izq	Entero	➡	Coordenada izquierda de la ventana
superior	Entero	➡	Coordenada superior de la ventana
tituloVentana	Cadena	➡	Título de la ventana del termómetro
textoTermo	Cadena	➡	Texto sobre el termómetro
cancelar	Cadena	➡	Texto del botón Cancelar
resultado	Entero	↩	Código de error

Nota versión 64 bits OS X

Este comando no es soportado en 4D Internet Commands a partir de la versión v16 R2 y debe ser eliminado de su código. Si se llama, devuelve el error -2201 (*función no implementada*).

IC IMAP Review Mail

- Comandos IMAP4, Introducción
- IMAP_Capability
- IMAP_CloseCurrentMB
- IMAP_CopyToMB
- IMAP_CreateMB
- IMAP_Delete
- IMAP_DeleteMB
- IMAP_Download
- IMAP_GetCurrentMB
- IMAP_GetFlags
- IMAP_GetMBStatus
- IMAP_GetMessage
- IMAP_GetPrefs
- IMAP_ListMBs
- IMAP_Login
- IMAP_Logout
- IMAP_MsgFetch
- IMAP_MsgInfo
- IMAP_MsgLst
- IMAP_MsgLstInfo
- IMAP_MsgNumToUID
- IMAP_RenameMB
- IMAP_Search
- IMAP_SetCurrentMB
- IMAP_SetFlags
- IMAP_SetPrefs
- IMAP_SubscribeMB
- IMAP_UIDToMsgNum
- IMAP_VerifyID

Comandos IMAP4, Introducción

El conjunto de comandos IMAP4 le permite a su base de datos acceder y manipular mensajes de correo electrónico en un servidor de correo IMAP y recuperar mensajes electrónicos de su servidor IMAP. Los comandos IMAP son compatibles con el protocolo Internet Message Access, versión 4 revisión 1 (IMAP4 rev1), definido por RFC 2060. IMAP4 rev1 permite la gestión de carpetas de mensajes a distancia, llamados "buzones", de la misma manera que los buzones locales.

Los comandos IMAP incluyen operaciones para crear, borrar y renombrar buzones, verificar la presencia de nuevos mensajes, eliminar permanentemente mensajes, asignar y borrar marcadores (flags) mensajes de búsqueda; y recuperar partes de mensajes.

Terminología

Conexión: se refiere a la secuencia completa de interacciones cliente/servidor IMAP, desde la conexión de red inicial (*IMAP_Login*) hasta el final de la conexión (*IMAP_Logout*).

Sesión: se refiere a la secuencia de interacciones cliente/servidor IMAP desde el momento en que se selecciona un buzón de correo (*IMAP_SetCurrentMB*) hasta el final de la conexión (*IMAP_SetCurrentMB*, *IMAP_CloseCurrentMB*) o hasta que se cierra la conexión (*IMAP_Logout*).

Presentación de las conexiones IMAP

- Inicialización de la comunicación TCP: *IT_MacTCPInit* (el comando *IT_PPPConnect* debe llamarse antes de *IT_MacTCPInit* en el caso de una conexión PPP).
- Apertura de una conexión: *IMAP_Login*
- Gestión de buzones: listar, crear, eliminar, renombrar, suscribirse o darse de baja, obtener el estado de los parámetros.
- Apertura de una sesión al definir el buzón de trabajo actual: *IMAP_SetCurrentMB*.
  Una vez se establece el buzón actual, puede administrar los mensajes que contiene.
- Gestión de mensajes: listar, descargar o eliminar mensajes; listar los marcadores de los mensajes, modificar los marcadores de los mensajes, copia a otro buzón, buscar y recuperar partes de mensajes sin descarga, etc.
- Una vez que haya terminado de trabajar con los mensajes del buzón actual, puede cerrar la sesión o abrir una nueva definiendo otro buzón actual. En todos los casos, el servidor IMAP actualiza de forma permanente sus mensajes. Por ejemplo, se eliminarán todos los mensajes con el marcador **\Deleted**.
- Una vez haya terminado, debe cerrar la sesión. Cierre de una conexión: *IMAP_Logout*.
- Otras operaciones: definición de las preferencias, capacidad, verificación de la conexión y reinicialización del temporizador de inactividad antes de cerrar la sesión automáticamente en el servidor IMAP.

Temas de comandos IMAP

Los comandos IMAP se dividen en dos secciones: **IC IMAP Review Mail** (intercambios con el servidor IMAP) e **IC Downloaded Mail** (tratamiento local). Estos comandos se han separado para mostrar los diferentes métodos de lectura de correo electrónico. Al leer el correo electrónico desde un servidor IMAP, los mensajes (o información de los mensajes) pueden importarse en las estructuras 4D (variables, campos, arrays) o descargarse en el disco. Esta sección describe las posibilidades ofrecidas por 4D Internet Commands para leer los mensajes desde un servidor IMAP.
El tamaño de los archivos a descargar determinan el uso de un modo u otro. Por ejemplo, un solo correo electrónico que contiene un archivo adjunto de 5 MB puede fácilmente superar la capacidad de almacenamiento de la base de datos. Sólo una imagen 4D o un campo BLOB es capaz de almacenar algo de este tamaño, sin embargo, la conversión de un mensaje o archivo adjunto a este formato no es muy eficiente ya que el acceso a la imagen o BLOB implica movilizar grandes recursos de memoria para acceder a la imagen o al BLOB. Para resolver este problema, el comando *IMAP_Download* transfiere los mensajes del servidor IMAP al disco duro del usuario.

Una vez importado al disco, la sección "**IC Downloaded Mail**" detalla los comandos que se utilizan para manipular archivos locales.

Mecanismos de buzones

Un buzón de correo IMAP puede ser manejado como una carpeta y puede contener archivos y/o subcarpetas. Del mismo modo, un buzón de correo puede contener mensajes y/o sub buzones.

Un buzón se accede utilizando su nombre jerárquico completo. Según el servidor IMAP, cada nivel jerárquico está separado por un separador de jerarquía (un separador se devuelve utilizando el comando *IMAP_ListMBs*).

Puede utilizar el separador para crear buzones hijos y para buscar los niveles más altos o más bajos de la jerarquía de nombres. Todos los hijos de un nivel jerárquico principal utilizan el mismo carácter separador

Nota: los mensajes sólo se pueden administrar una vez se ha seleccionado el buzón de trabajo actual (*IMAP_SetCurrentMB*).

Cada cuenta puede tener uno o varios buzones.

Los nombres de buzones tienen en cuenta las mayúsculas y minúsculas, por lo tanto, no se pueden crear dos buzones con nombres que difieran sólo por tener mayúsculas o minúsculas.

El buzón INBOX es un caso particular: existe en todas las cuentas y se utiliza para almacenar los mensajes recibidos. El INBOX se crea automáticamente cada vez que una cuenta se configura.

Un usuario no puede eliminar el INBOX pero sí puede cambiar su nombre. Si decide cambiar el nombre, se crea inmediatamente un nuevo INBOX vacío. El nombre INBOX no tiene en cuenta las mayúsculas y minúsculas.

Algunos atributos del buzón, como el número total de mensajes o de mensajes nuevos, pueden obtenerse, incluso si el buzón no es el actual.

numMsg e IDunico

Para el uso de los comandos IMAP, es importante entender completamente los parámetros *numMsg* e *IDunico*.

numMsg es el número de un mensaje en el buzón en el momento que se ejecuta el comando *IMAP_SetCurrentMB*.

Una vez seleccionado un buzón actual, los mensajes en el buzón se numeran del 1 hasta el número total de elementos en el buzón. Los números se asignan en función del orden en que los mensajes fueron recibidos en el buzón, siendo 1 el más antiguo. Los números asignados a los mensajes sólo son válidos desde el momento de seleccionar el buzón de trabajo actual (*IMAP_SetCurrentMB*) hasta que se cierra (*IMAP_CloseCurrentMB*, *IMAP_SetCurrentMB* o *IMAP_Logout*).

Cuando el buzón se cierra, todos los mensajes marcados para eliminación se borran

Cuando el usuario inicia sesión de nuevo en el servidor IMAP, los mensajes presentes en el buzón se numeran de nuevo del 1 al X. Por ejemplo, si hay 10 mensajes en el buzón y si los mensajes del 1 al 5 se eliminan cuando el usuario vuelve a abrir el buzón, los antiguos mensajes 6-10 se han vuelto los mensajes del 1 al 5.

Por ejemplo, veamos el siguiente ejemplo: conéctese a un servidor IMAP y obtenga la siguiente lista de mensajes:

numMsg	IDunico	Fecha	De	Asunto
1	10005	1 Jul 2001 ...	danw@acme.com	Clientes potenciales...
2	10008	1 Jul 2001 ...	frank@acme.com	Orden de licencia en sitio
3	10012	3 Jul 2001 ...	joe@acme.com	¿Alguien quiere ir a almorzar?
4	20000	4 Jul 2002 ...	kelly@acme.com	Su esposa llamó...
5	20001	4 Jul 2002 ...	track@fedex.com	Seguimiento FedEx

Durante esta sesión, usted borra los mensajes 3 y 4. Al cerrar el buzón de trabajo actual, se realizan las eliminaciones. Cuando vuelva al servidor, la lista de mensajes se reenumerará así:

numMsg	IDunico	Fecha	De	Asunto
1	10005	1 Jul 2001 ...	danw@acme.com	Clientes potenciales...
2	10008	1 Jul 2001 ...	frank@acme.com	Orden de licencia en sitio
3	20001	4 Jul 2002 ...	track@fedex.com	Seguimiento FedEx

numMsg no es un valor estático y pueden variar de una sesión a otra. Cambiará en relación con otros mensajes en el buzón en el momento en que se selecciona el buzón de trabajo actual.

Por el contrario, *IDunico* es un número único asignado al mensaje por el servidor IMAP en un orden estrictamente ascendente. A medida que cada mensaje se agrega al buzón, se le asigna un identificador más alto que el del mensaje previamente añadido.

Desafortunadamente, los servidores IMAP no utilizan el IDunico como referencia principal para sus mensajes. Al utilizar los comandos IMAP tendrá que especificar el *numMsg* como referencia para los mensajes en el servidor. Los desarrolladores deben tener cuidado al desarrollar soluciones que referencian los mensajes en la base de datos, dejando el cuerpo del mensaje en el servidor.

Recomendaciones

Como la característica principal de IMAP es la interoperabilidad, la recomendación final es "Probar TODO." Es recomendable, probar el cliente con todos los servidores en los que tiene cuenta.

Para mayor información, consulte los siguientes sitios:

- IMAP Products and Services: *http://www.imap.org/products.html*
- MailConnect: *http://www.imc.org/imc-mailconnect*.

Comparación de los comandos POP3 e IMAP4

Login	Equivalente	No parámetro POP para IMAP
VerifyID	Equivalente	
Delete	Equivalente	Los comandos IMAP borran en tiempo real. Los comandos POP3 requieren POP3_Logout para eliminar los mensajes de forma permanente. IMAP_SetFlags con el marcador \Deleted permite obtener el mismo resultado que POP3_Delete
Logout	Equivalente	
SetPrefs	Equivalente	Sin *carpetaDocAdj* para IMAP, el parámetro POP3 *carpetaDocAdj* se vuelve opcional
GetPrefs	Equivalente	Ver nota *carpetaDocAdj* en SetPrefs
MsgLstInfo	Equivalente	
MsgInfo	Equivalente	
MsgLst	Equivalente	
UIDToMsgNum	Equivalente	unicoID es un Entero largo para IMAP y una cadena para POP3
Download	Equivalente	
POP3_Reset	No direct equiv	Necesita la combinación de IMAP_Search en los marcadores \Deleted e IMAP_SetFlags para borrar los marcadores \Deleted
POP3_BoxInfo	No equivalencia directa	Necesita la combinación de los comandos IMAP_SetCurrentMB e IMAP_MsgLstInfo
IMAP_ MsgNumToUID	No equivalencia directa	
GetMessage	Casi equivalente	IMAP es más poderoso ya que permite seleccionar únicamente el cuerpo del mensaje
POP3_Charset	No equivalente	IMAP maneja automáticamente los conjuntos de caracteres
IMAP_Capability	No equivalente	Específico al protocolo IMAP
IMAP_ListMBs	No equivalente	Específico al protocolo IMAP
IMAP_GetMBStatus	No equivalente	Específico al protocolo IMAP
IMAP_SetCurrentMB	No equivalente	Específico al protocolo IMAPl
IMAP_GetCurrentMB	No equivalente	Específico al protocolo IMAP
IMAP_CloseCurrentMB	No equivalente	Específico al protocolo IMAP
IMAP_CopyToMB	No equivalente	Específico al protocolo IMAP
IMAP_SubscribeMB	No equivalente	Específico al protocolo IMAP
IMAP_CreateMB	No equivalente	Específico al protocolo IMAP
IMAP_DeleteMB	No equivalente	Específico al protocolo IMAP
IMAP_RenameMB	No equivalente	Específico al protocolo IMAP
IMAP_SetFlags	No equivalente	Específico al protocolo IMAP
IMAP_GetFlags	No equivalente	Específico al protocolo IMAP
IMAP_Search	No equivalente	Específico al protocolo IMAP
IMAP_MsgFetch	No equivalente	Específico al protocolo IMAP

Notas:

- Servidores IMAP y POP3: en el caso de los servidores IMAP, no declare *msgID* de la misma forma, dado que *msgID* es un entero largo.
- La eliminación no funciona exactamente de la misma manera entre los protocolos POP3 e IMAP. *IMAP_Delete* elimina los mensajes en tiempo real. Para obtener el mismo resultado que *POP3_Delete*, use *IMAP_SetFlags* para establecer el marcador **\Deleted**; para obtener el mismo resultado que *POP3_Reset*, use *IMAP_SetFlags* para recuperar los marcadores **\Deleted**.
- Para una mayor flexibilidad, los comandos de Internet de 4D pueden omitir una referencia a la conexión IMAP controles de bajo nivel TCP y viceversa. Para obtener más información, consulte la sección **Rutinas de bajo nivel, Presentación**.

IMAP_Capability

IMAP_Capability (imap_ID ; compatibilidades) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	➡	Referencia de conexión IMAP
compatibilidades	Texto	⬅	Compatibilidades IMAP
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_Capability* devuelve un valor de Texto que contiene la lista de las diferentes compatibilidades del servidor IMAP. La lista de compatibilidades indica la versión de IMAP así como también las funciones opcionales (como extensiones, revisiones o correcciones al protocolo IMAP4rev1), aceptadas por el servidor.

La compatibilidad IMAP4rev1 debe aparecer en el texto de compatibilidad con el fin de asegurar el buen funcionamiento del servidor con 4D Internet Commands.

IMAP_CloseCurrentMB

IMAP_CloseCurrentMB (imap_ID) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	→	Referencia de conexión IMAP
resultado	Entero	←	Código de error

Descripción

El comando *IMAP_CloseCurrentMB* cierra el buzón de trabajo actual sin seleccionar otro buzón ni ejecutar *IMAP_Logout*. *IMAP_CloseCurrentMB* elimina permanentemente todos los mensajes que tienen el marcador **\Deleted**.

Nota: IMAP permite a los usuarios trabajar simultáneamente con el mismo buzón de correo en modo cliente/servidor. Cuando un usuario realiza la sincronización y mantiene la conexión abierta, el buzón utilizado por última vez permanecerá en el modo seleccionado. Si otro usuario trata de utilizar este buzón no obtendrá información válida, o no podrá trabajar correctamente, dependiendo de la implementación del servidor, incluso si el usuario trabaja en modo "desconectado" (es decir, conectado, pero trabajando con los datos).

imap_ID es una referencia entero largo a una conexión abierta creada con *IMAP_Login*.

IMAP_CopyToMB

IMAP_CopyToMB (imap_ID ; primerMsg ; ultimoMsg ; nomBuzonObjetivo ; borrarMsg) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	→	Referencia de conexión IMAP
primerMsg	Entero largo	→	Número del primer mensaje
ultimoMsg	Entero largo	→	Número del último mensaje
nomBuzonObjetivo	Texto	→	Nombre del buzón de destino
borrarMsg	Entero	→	0= No eliminar del buzón fuente, 1= Eliminar del buzón fuente
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_CopyToMB* copia los mensajes en el intervalo de *primerMsg* a *ultimoMsg* al final del buzón de destino *nomBuzonObjetivo*. Los marcadores y fechas internas de los mensajes por lo general se conservan en el buzón de destino, en función de la implementación del servidor IMAP.

Después de la copia, los mensajes originales no se eliminan del buzón fuente. Si desea eliminarlos, puede utilizar uno de los siguientes tres procesos:

- Utilizar el comando *IMAP_Delete*,
- Pasar el valor 1 en el parámetro opcional *borrarMsg*,
- ubicar el marcador *IMAP_SetFlags* (**\Deleted**): los mensajes se eliminan cuando se cierra la sesión.

Nota: el parámetro *borrarMsg* obliga la ejecución de *IMAP_Delete*; por lo tanto, la eliminación incluirá los mensajes entre *primerMsg* y *ultimoMsg* y TODOS los mensajes con el marcador **\Deleted**.

Si el buzón de destino no existe, se devuelve un error.

imap_ID es una referencia entero largo a una conexión abierta creada con *IMAP_Login*.

primerMsg es un número entero largo que especifica el número del primer mensaje a copiar. El número de mensaje es un valor que representa la posición de un mensaje en la lista de todos los mensajes en el buzón identificado por *imap_ID*.

ultimoMsg es un número entero largo que indica el número del último mensaje a copiar. El número de mensaje es un valor que representa la posición de un mensaje en la lista de todos los mensajes en el buzón identificado por *imap_ID*.

Nota: los comandos *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* y *IMAP_CopyToMB* no devuelven un error si *primerMsg* es mayor que *ultimoMsg*. En caso de que esto ocurra, el comando no hace nada.

nomBuzonObjetivo es el nombre completo del buzón donde se van a copiar los mensajes.

El parámetro opcional *borrarMsg* permite indicar si desea eliminar el mensaje del buzón de origen.
• 0= No eliminar los mensajes (valor por defecto);
• 1= Borrar los mensajes.

Si se omite *borrarMsg* se utiliza el valor por defecto.

Si la copia falla, el mensaje no se elimina del buzón de origen.

Si el usuario no tiene derechos de acceso suficientes para eliminar los mensajes, se genera un mensaje de error.

IMAP_CreateMB

IMAP_CreateMB (imap_ID ; nomBuzon) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	➡	Referencia de conexión IMAP
nomBuzon	Texto	➡	Nombre del buzón creado
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_CreateMB* crea un buzón con el nombre dado. Si el separador de jerarquía del servidor IMAP aparece en otra parte en el nombre del buzón, el servidor IMAP creará los buzones padres necesarios para la creación del buzón especificado.

Por ejemplo, si la creación de la caja "Proyectos/IMAP/Doc" se solicita en un servidor en el que "/" es el separador jerárquico:

- Sólo se crea el buzón "Doc" si "Proyectos" e "IMAP" ya existen.
- Los buzones "IMAP" y "Doc" se crean sólo si "Proyectos" ya existe.
- Los buzones "Proyectos", "IMAP" y "Doc" se crean si no existen.

imap_ID es una referencia entero largo a una conexión abierta creada con *IMAP_Login*.

nomBuzon es el nombre completo del buzón a crear (ver las reglas de nombres en la introducción de la sección IMAP).

Nota: el intento de crear un buzón llamado INBOX (que es un nombre especial reservado que significa "buzón principal para este usuario en este servidor") o un buzón con un nombre que se refiere a un buzón existente provocará un error.

IMAP_Delete

IMAP_Delete (imap_ID ; primerMsg ; ultimoMsg) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	➡	Referencia de conexión IMAP
primerMsg	Entero largo	➡	Número del primer mensaje
ultimoMsg	Entero largo	➡	Número del último mensaje
resultado	Entero	⮌	Código de error

Descripción

El comando *IMAP_Delete* ubica el marcador **\Deleted** en cada mensaje del intervalo comprendido entre *primerMsg* y *ultimoMsg*, luego borra todos los mensajes que tienen este marcador (incluidos los mensajes en los que el marcador **\Deleted** haya sido fijado previamente para la sesión actual). La eliminación es efectuada por el servidor IMAP y se lleva a cabo cuando se cierra la conexión (*IMAP_Logout*) o al seleccionar otro buzón actual (*IMAP_SetCurrentMB*) o al cerrar el buzón actual (*IMAP_CloseCurrentMB*).

Si no desea que la eliminación sea inmediata, puede utilizar el comando *IMAP_SetFlags* y configurar el marcador **\Deleted** para borrar los mensajes después.

imap_ID es una referencia entero largo a una conexión abierta creada con *IMAP_Login*.

El parámetro *primerMsg* es el número del primer mensaje a eliminar.

El parámetro *ultimoMsg* es el número del último mensaje a eliminar.

Nota: los comandos *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* e *IMAP_CopyToMB* no devuelven un error si *primerMsg* es mayor que *ultimoMsg*. En caso de que esto ocurra, el comando no hace nada.

IMAP_DeleteMB

IMAP_DeleteMB (imap_ID ; nomBuzon) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	➡	Referencia de conexión IMAP
nomBuzon	Texto	➡	Nombre del buzón a borrar
resultado	Entero	⮌	Código de error

Descripción

El comando *IMAP_DeleteMB* elimina de forma permanente el buzón especificado por nomBuzon. El intento de eliminar un buzón INBOX o un buzón que no existe producirá un error.

El comando *IMAP_DeleteMB* no puede eliminar un buzón que tiene sub buzones y el atributo de buzón **\Noselect** .

Es posible eliminar un buzón que tiene sub buzones y el atributo **\Noselect**. En este caso, todos los mensajes en el buzón se eliminan y se aplica el atributo **\Noselect**.

Nota: el protocolo IMAP no garantiza que pueda eliminar un buzón que no está vacío, aunque en algunos servidores esto está permitido. Si decide intentarlo, debe estar preparado para utilizar otro método si el más conveniente falla. Además, no debe tratar de eliminar el buzón de trabajo actual mientras esté abierto, primero debe cerrarlo, algunos servidores no permiten la eliminación del buzón actual.

imap_ID es una referencia entero largo de una conexión abierta creada con *IMAP_Login*.

nomBuzon es el nombre completo del buzón a eliminar.

IMAP_Download

IMAP_Download (imap_ID ; numMsg ; encabSolo ; nomArchivo ; actSeen) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	→	Referencia de conexión IMAP
numMsg	Entero largo	→	Número del mensaje
encabSolo	Entero	→	0 = Mensaje entero, 1 = Encabezado únicamente
nomArchivo	Texto	→	Nombre del archivo local
		←	Nombre del archivo local utilizado
actSeen	Entero	→	0 = Añadir marcador \Seen; 1= No añadir el marcador \Seen
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_Download* está diseñado para recuperar un mensaje de un servidor IMAP, grabándolo en el disco local. Todo mensaje IMAP que contiene archivos adjuntos o cuyo tamaño es mayor a 32 K debe ser descargado con este comando. Los archivos adjuntos sólo pueden extraerse de los mensajes recuperados de esta manera.

imap_ID es una referencia entero largo a una conexión abierta creada con *IMAP_Login*.

numMsg es un valor entero largo que indica cual mensaje en el buzón recuperar. *numMsg* representa la posición de un mensaje dentro de la lista actual de mensajes. El *numMsg* de un mensaje no es un valor estable, difiere de una sesión a otra.

encabSolo es un valor entero que indica si se debe recuperar todo el mensaje o sólo la información del encabezado.

nomArchivo contiene el nombre del archivo y la ruta de acceso opcional donde desea guardar el mensaje. Este valor se puede especificar de tres formas diferentes:

- "" = Guarda el archivo en la carpeta definida por *IMAP_SetPrefs*, con el nombre "temp1" (si un archivo con el mismo nombre ya existe, los nombres de archivo "temp2", "temp3", etc. se utilizarán hasta que se encuentre un nombre de archivo sin utilizar).
- "NomArchivo" = Guarda el archivo en la carpeta definida por *IMAP_SetPrefs* con el nombre nomArchivo.
- "Ruta:nomArchivo" = Guarda el archivo en la ruta especificada con el nombre nomArchivo.

En los dos primeros casos, si *IMAP_SetPrefs*, no especifica ninguna carpeta, el mensaje se guardará en la misma carpeta que la estructura de la base de datos (con 4D monopuesto) o en la carpeta 4D Client (con 4D Server).

Después de que el archivo se guarde en el disco, el nombre final del archivo se devuelve en el parámetro *nomArchivo*. Si intenta llamar a *IMAP_Download* con un nombre de archivo que ya existe dentro de la carpeta de descarga, el nombre se incrementa numéricamente y su nuevo valor como se guarda en el disco será devuelto a la variable *nomArchivo*.

actSeen es un valor entero que indica si el marcador **\Seen** se debe agregar a la lista de marcadores del mensaje. Este parámetro es opcional y se utiliza un valor por defecto si no se pasa este parámetro:

- 0 = Añadir el marcador **\Seen**
- 1 = No añadir el marcador **\Seen**

El valor por defecto es 0 lo que significa implícitamente, añadir el marcador **\Seen**.

IMAP_GetCurrentMB

IMAP_GetCurrentMB (imap_ID ; nomBuzon) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	➡	Referencia de conexión IMAP
nomBuzon	Texto	⬅	Nombre del buzón actual
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_GetCurrentMB* devuelve el nombre del buzón actual.

imap_ID contiene la referencia de una sesión abierta con *IMAP_Login*.

nomBuzon devuelve el nombre completo del buzón actual. Si *nomBuzon* devuelve una cadena vacía, ningún buzón actual está seleccionado.

IMAP_GetFlags

IMAP_GetFlags (imap_ID ; primerMsg ; ultimoMsg ; arrayMarcMsg ; arrayNumMsg) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	→	Referencia de conexión IMAP
primerMsg	Entero largo	→	Número del primer mensaje
ultimoMsg	Entero largo	→	Número del último mensaje
arrayMarcMsg	Array cadena	←	Valores de los marcadores para cada mensaje
arrayNumMsg	Array entero largo	←	Array de los números de mensajes
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_GetFlags* devuelve la lista de marcadores para los mensajes especificados.

imap_ID es una referencia entero largo a una conexión abierta creada con *IMAP_Login*.

primerMsg es un número entero largo que especifica el número del primer mensaje a examinar. El número de mensaje es un valor que representa la posición de un mensaje en la lista de todos los mensajes en el buzón de trabajo actual.

ultimoMsg es un número entero largo que indica el número del último mensaje a examinar. El número de mensaje es un valor que representa la posición de un mensaje en la lista de todos los mensajes en el buzón de trabajo actual.

Nota: los comandos *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* y *IMAP_CopyToMB* no devuelven un error si *primerMsg* es mayor que el *ultimoMsg*. En el caso de que esto ocurra, el comando no hace nada.

arrayMarcMsg recibe la lista de marcadores, separados por espacios, de cada mensaje cuyo número está entre *primerMsg* y *ultimoMsg.*

arrayNumMsg devuelve los números de los mensajes entre *primerMsg* y *ultimoMsg*.

IMAP_GetMBStatus

IMAP_GetMBStatus (imap_ID ; nomBuzon ; numMsg ; numNuevMsg ; numMsgNoLeido ; unicoIDB) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	→	Referencia de conexión IMAP
nomBuzon	Texto	→	Nombre del buzón
numMsg	Entero largo	←	Número de mensajes en el buzón especificado
numNuevMsg	Entero largo	←	Número de mensajes con el marcador \Recent
numMsgNoLeido	Entero largo	←	Número de mensajes sin el marcador \Seen
unicoIDB	Entero largo	←	Número de identificación único del buzón especificado
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_GetMBStatus* devuelve el estado de los parámetros del buzón especificado por *nomBuzon*. No cambia el buzón actual (ver *IMAP_SetCurrentMB*), ni afecta el estado de los mensajes en el buzón especificado (en particular, por lo general no borra los marcadores **\Recent**, pero esto puede variar en función de los parámetros del servidor IMAP4). Este comando alternativo permite obtener el estado de los parámetros sin desactivar el buzón actual.

Este comando es particularmente útil para:

- Verificar o recuperar el número de identificación único de un buzón,
- Verificar los mensajes recientes o no leídos de un buzón sin abrir una sesión especifica.

Importante: le recomendamos no llamar el comando *IMAP_GetMBStatus* utilizando el buzón actual. De esta manera, puede encontrar problemas y la información devuelta puede no estar sincronizada con el estado del buzón actual (en particular para los nuevos e-mails).

imap_ID es una referencia entero largo a una conexión abierta creada con *IMAP_Login*.

nomBuzon es el nombre completo del buzón existente para el que desea obtener los parámetros de estado.

Nota: a diferencia del comando *IMAP_ListMBs*, el parámetro *nomBuzon* no acepta comodines.

numMsg devuelve el número de mensajes en el buzón actual (devuelve -1 en caso de error).

numNuevMsg devuelve el número de mensajes recientes en el buzón actual (devuelve -1 en caso de error).

numMsgNoLeido devuelve el número de mensajes no leídos en el buzón actual (devuelve -1 en caso de error)

unicoIDB devuelve el número de identificación único del buzón (devuelve -1 en caso de error).

Con el protocolo IMAP4, el nombre del buzón no es suficiente para identificar a un buzón. Por lo tanto, un identificador único valor se asocia a cada buzón. Este identificador es particularmente valioso para la sincronización de tareas.
De esta forma, puede comprobar si el buzón "A" ha sido renombrado como "B" o eliminado, simplemente verificando su número de identificación único.
Por otra parte, este identificador permite comprobar si un buzón llamado "A" se ha eliminado y si se ha creado otro buzón "A".

IMAP_GetMessage

IMAP_GetMessage (imap_ID ; numMsg ; offset ; longitud ; parteMsg ; textoMsg {; actSeen}) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	→	Referencia de conexión IMAP
numMsg	Entero largo	→	Número del mensaje
offset	Entero largo	→	Carácter a partir del cual comenzar la recuperación
longitud	Entero largo	→	Número de caracteres a reenviar
parteMsg	Entero	→	0 = Mensaje entero, 1 = Encabezado únicamente, 2 = Cuerpo únicamente
textoMsg	Texto	←	Texto del mensaje
actSeen	Entero	→	0 = Actualizar marcador \Seen; 1= No actualizar el marcador \Seen
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_GetMessage* devuelve el texto completo del mensaje identificado por *msgNum* en el buzón actual definido por *IMAP_SetCurrentMB*. A menos que se especifique lo contrario por el comando *IMAP_SetPrefs*, se eliminarán los caracteres de salto de línea en el mensaje.

El comando *IMAP_GetMessage* devuelve todo el mensaje, incluyendo la información del encabezado o únicamente una parte del mensaje (encabezado o cuerpo) en función del valor del parámetro *parteMsg*.

imap_ID es una referencia entero largo a una conexión abierta creada con *IMAP_Login*.

numMsg es un valor entero largo que indica cual mensaje en el buzón recuperar. Este número representa la posición de un mensaje dentro de la lista actual de los mensajes. Atención, el *numMsg* de un mensaje no es un valor estable, difiere de una sesión a otra.

offset es un valor entero largo que indica la posición del caracter a partir del cual comenzar la lectura. En la mayoría de los casos pase cero en este parámetro.

El parámetro *longitud* es un valor entero largo que indica el número de caracteres a recuperar más allá de la posición *offset*.

parteMsg indica la parte del mensaje a recuperar. Se pueden pasar los valores 0, 1 o 2:

- 0 = Todo el mensaje,
- 1 = Sólo el encabezado,
- 2 = Cuerpo únicamente (es decir a partir del primer Texto/plano encontrado).

Cuando recupera el mensaje entero o el encabezado únicamente, recupera texto bruto no decodificado. Por otra parte, al recuperar sólo el cuerpo, el texto obtenido se decodifica automáticamente y si es necesario se convierte (ver *POP3_Charset* para obtener más información sobre las reglas de decodificación y conversión).

actSeen es un valor entero que indica si el marcador **\Seen** debe añadirse o no a los marcadores del mensaje. Este parámetro es opcional y se utiliza el valor por defecto si no se pasa este parámetro.

- 0 = Añadir el marcador **\Seen** (valor por defecto);
- 1= No añadir el marcador **\Seen**;

textoMsg es una variable texto que recibe el texto recuperado.

IMAP_GetPrefs

IMAP_GetPrefs (retornoLinea ; carpetaMsg) -> resultado

Parámetro	Tipo		Descripción
retornoLinea	Entero	⇐	0 = No retirar los retornos de línea, 1 = Retirar los retornos de línea, -1 = No cambiar
carpetaMsg	Texto	⇐	Ruta de acceso a la carpeta de mensajes
resultado	Entero	⇒	Código de error

Descripción

El comando *IMAP_GetPrefs* devuelve las preferencias actuales para los comandos IMAP.

Las preferencias se devuelven en las variables listadas en los parámetros.

El parámetro *retornoLinea* devuelve el parámetro actual de la opción de eliminación de los retornos de línea.

El parámetro *carpetaMsg* devuelve la ruta de acceso local de la carpeta donde se guardan por defecto los mensajes recuperados.

IMAP_ListMBs

IMAP_ListMBs (imap_ID ; refBuzon ; nomBuzon ; arraynomsbuzon ; arrayAtribbuzon ; arraySepJerarq ; buzonesInscritos) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	→	Referencia de conexión IMAP
refBuzon	Texto	→	Cadena vacía o nombre de buzón o nivel jerárquico
nomBuzon	Texto	→	Cadena vacía o nombre del buzón o Jokers
arraynomsbuzon	Array cadena	←	Array de nombres de buzones (rutas de acceso)
arrayAtribbuzon	Array cadena	←	Array de atributos de buzones
arraySepJerarq	Array cadena	←	Array de separadores jerárquicos
buzonesInscritos	Entero	→	0 = Lista todos los buzones disponibles 1 = Lista únicamente los buzones inscritos
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_ListMBs* devuelve la lista de buzones disponibles para el usuario conectado y la información adjunta. Si este comando falla, se inicializan los arrays especificados.

refBuzon y *nomBuzon* deben ser considerados conjuntamente ya que la lista de buzones resultante dependerá de la combinación de los valores de estos parámetros.

Cuando pasa 1 en el último parámetro, *buzonesInscritos*, la lista devuelta puede estar restringida a los buzones a los cuales el usuario está suscrito (ver *IMAP_SubscribeMB*).

Si la ejecución de *IMAP_ListMBs* toma mucho tiempo, ya sea por un gran número de buzones a examinar o por las numerosas y complejas estructuras jerárquicas, etc., puede:

- utilizar comodines (ver más adelante) con *IMAP_ListMBs*,
- pasar 1 como parámetro *buzonesInscritos*, para listar únicamente los buzones definidos con el comando *IMAP_SubscribeMB*.

imap_ID es una referencia entero largo a una conexión abierta creada con *IMAP_Login*.

refBuzon es un valor texto que debe ser combinado con el parámetro *nomBuzon* para determinar los buzones a utilizar. La referencia (*refBuzon*) debe utilizarse como un directorio de trabajo actual en los sistemas Unix. En otras palabras, el nombre del buzón (*nomBuzon*) se interpreta como un archivo ubicado en el directorio especificado por la referencia (*refBuzon*). las especificaciones IMAP indican que la interpretación de la referencia (*refBuzon*) "depende de la implementación", esto significa que no es obligatoria. Es recomendable que el usuario reciba un modo de funcionamiento que no utilice ningún argumento de referencia *refBuzon*. Como tal, puede interactuar con servidores más antiguos que no implementaron el uso de argumentos de referencia.

Si *refBuzon* es una cadena vacía, sólo el parámetro *nomBuzon* se utilizará para listar los buzones.

Si *refBuzon* contiene el nombre de un buzón o un nivel jerárquico de buzones, debe ser usado para definir el contexto en el que debe ser interpretado el parámetro *nomBuzon*.

Nota: recomendamos colocar un separador jerárquico en el argumento de referencia cuando se utilice. Esto garantizará la compatibilidad del comando independientemente del servidor IMAP utilizado.

El valor a pasar en el parámetro *nomBuzon* depende del parámetro *refBuzon*.

Si *nomBuzon* es una cadena vacía, se devuelve el separador jerárquico.

Nota: si decide implementar un sistema de conexiones múltiples con la ayuda del parámetro *refBuzon*, debe permitir al usuario elegir si desea o no utilizar un separador jerárquico al inicio del nombre del buzón. La gestión del separador al inicio del nombre varía de un servidor a otro, e incluso entre diferentes sistemas de correo electrónico en el mismo servidor. En algunos casos, este separador jerárquico significará "no tener en cuenta el argumento de referencia", mientras que en otros casos, los dos parámetros se concatenan y el carácter separador se ignora.

El array *arrayNomBuzon* recibe la lista de nombres de buzones disponibles.

El array *arrayAtribBuzon* recibe la lista de atributos de los buzones disponibles.

Atributos de buzones
Hay cuatro atributos disponibles:

- **\Noinferiors**: no existen niveles jerárquicos inferiores y ni pueden crearse.
- **\Noselect**: este nombre no puede utilizarse como un buzón seleccionable.
- **\Marked**: el servidor ha marcado el buzón como "interesante", probablemente el buzón contiene mensajes agregados desde la última selección.
- **\Unmarked**: el buzón no contiene ningún mensaje adicional desde la última selección.

El array *arraySepJerar* recibe la lista de separadores jerárquicos de los buzones disponibles.

El separador jerárquico es un carácter utilizado para delimitar los niveles jerárquicos en un nombre de buzón. Puede utilizar este carácter para crear buzones hijos y para buscar los niveles más altos o más bajos de la jerarquía de nombres. Todos los hijos de un nivel jerárquico principal utilizan el mismo carácter separador.

buzonesInscritos es un valor entero que permite indicar si quiere recuperar la lista de buzones a los cuales el usuario está suscrito. Un valor cero lista todos los buzones usuario disponibles. Un valor de 1 lista únicamente los buzones usuario suscritos. *buzonesInscritos* es un parámetro opcional que si no se define tiene el valor de cero por defecto.

Ejemplo 1

El siguiente ejemplo:

```
IMAP_ListMBs(imap_ID;"4DIC/Work/";"Test";mbNamesArray;mbAttribsArray;mbHierarArray)
```

...devuelve todos los buzones disponibles del buzón “4DIC/Work/Test”.
Recuerde que si el servidor IMAP no interpreta los parámetros como lo desea, no utilice el parámetro *refBuzon* y concatene los valores de *refBuzon y nomBuzon* en el parámetro *nomBuzon*:

```
IMAP_ListMBs(imap_ID;"";"4DIC/Work/Test";mbNamesArray;mbAttribsArray;mbHierarArray)
```

Ejemplo 2

El siguiente ejemplo:

```
IMAP_ListMBs(imap_ID;"";"";mbNamesArray;mbAttribsArray;mbHierarArray)
```

...devuelve el separador jerárquico.

Uso de comodines

Puede utilizar comodines en los parámetros *refBuzon* y *nomBuzon* con el fin de facilitar la selección del buzón. A continuación encontrará un ejemplo que utiliza los comodines más comunes, sin embargo tenga en cuenta que la interpretación de los comodines depende del servidor IMAP; por lo tanto, estos ejemplos podrían no funcionar. En este caso, verifique los comodines de su servidor IMAP.

- “ * ” reemplaza todo carácter en su posición:

```
IMAP_ListMBs(imap_ID;"";"*";mbNamesArray;mbAttribsArray;mbHierarArray)
```

... devuelve todos los buzones disponibles para el usuario conectado.

```
IMAP_ListMBs(imap_ID;"";"Work*";mbNamesArray;mbAttribsArray;mbHierarArray)
```

... devuelve todos los buzones disponibles que comiencen por la raíz “Work”.

- “ % ” es similar a “ * ”, pero no remplaza el separador jerárquico. Si el comodín “%” es el último carácter del parámetro *nomBuzon*, los niveles jerárquicos correspondientes también se devuelven.

Si estos niveles jerárquicos no son buzones seleccionables, se devuelven con el atributo **\Noselect** (ver el párrafo "Atributos de buzones").

```
IMAP_ListMBs(imap_ID"";"Work/%";mbNamesArray;mbAttribsArray;mbHierarArray)
```

... devuelve todos los buzones disponibles que comienzan por la raíz "Work", más un subnivel jerárquico adicional.

El comodín "%" puede ser útil durante el análisis por nivel de la jerarquía de los buzones.
Dada la siguiente jerarquía de buzones:

```
INBOX
    MailboxA
        MailboxAA
        MailboxAB
    MailboxB
        MailboxBA
        MailboxBB
    MailboxC
        MailboxCA
        MailboxCB
```

```
IMAP_ListMBs(imap_ID;"";"%";mbNamesArray;mbAttribsArray;mbHierarArray)
```

... devuelve INBOX, MailboxA, MailboxB y MailboxC.

```
IMAP_ListMBs(imap_ID;"";"MailboxA%";mbNamesArray;mbAttribsArray;mbHierarArray)
```

... devuelve MailboxAA y MailboxAB.

Utilizando esta técnica, puede dar al usuario una cierta flexibilidad sin devolver demasiada información

```
IMAP_ListMBs(imap_ID;"";"*";mbNamesArray;mbAttribsArray;mbHierarArray)
```

Note que los servidores IMAP pueden limitar el número de niveles a analizar.

IMAP_Login

IMAP_Login (nomServidor ; nomUsuario ; contraseña ; imap_ID {; paramSesion}) -> resultado

Parámetro	Tipo		Descripción
nomServidor	Cadena	➡	Nombre o dirección IP del servidor de correo IMAP
nomUsuario	Cadena	➡	Nombre del usuario
contraseña	Cadena	➡	Contraseña
imap_ID	Entero largo	⬅	Referencia a esta conexión IMAP
paramSesion	Entero largo	➡	1 = Use SSL, 0 se omite = No utilizar SSL
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_Login* conecta al usuario al servidor de correo IMAP con el nombre de usuario y contraseña dados.

IMAP_Login devuelve un número de identificación específico para la conexión (*imap_ID*) al cual los comandos IMAP posteriores pueden referirse.

La conexión se cierra con el comando *IMAP_Logout* o cuando el contador de inactividad del servidor IMAP supera el timeout.

nomServidor es el nombre o la dirección IP del servidor de correo IMAP. Se recomienda utilizar el nombre del servidor, si es necesario, puede utilizar una dirección IP.

nomUsuario es el nombre del usuario del servidor de correo IMAP. El parámetro *nomUsuario* no debe contener el dominio. Por ejemplo, para la dirección "jack@4d.com", el *nomUsuario* es “jack”.

contraseña es la contraseña correspondiente a *nomUsuario* en el servidor de correo IMAP.

imap_ID es una variable de tipo entero largo en la que se devuelve una referencia a la conexión que se acaba de establecer. Este parámetro debe pasarse a una variable 4D para aceptar los resultados devueltos. La variable se utiliza en todos los comandos posteriores que realizan acciones relacionadas con esta sesión. Si *IMAP_Login*, *imap_ID* toma el valor cero.

El parámetro opcional *paramSesion* permite activar el protocolo SSL para la conexión:

- Si pasa 1, la conexión al servidor IMAP se efectúa en SSL (modo síncrono),
- Si pasa 0 u omite este parámetro, la conexión se efectúa en modo estándar, no seguro.

Ejemplo

Esta es una secuencia típica de conexión:

```
$ErrorNum:=IMAP_Login(vHost;vUserName;vUserPassword;vImap_ID;1)
If($ErrorNum =0)
   C_TEXT(vCapability)
   $ErrorNum:=IMAP_Capability(vImap_ID;vCapability))
  ` Los comandos IMAP utilizan el parámetro vImap_ID
End if
$ErrorNum:=IMAP_Logout(vImap_ID)
```

IMAP_Logout

IMAP_Logout (imap_ID) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	➡	Referencia de conexión IMAP
		⬅	0 = Conexión cerrada
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_Logout* cierra la conexión IMAP referenciada por la variable *imap_ID*.
Si la desconexión del servidor IMAP se efectúa correctamente, el valor 0 (cero) se devuelve en *imap_ID*.

Nota: al cerrar la conexión automáticamente se cierra la sesión actual.

IMAP_MsgFetch

IMAP_MsgFetch (imap_ID ; numMsg ; datosMsg ; valoresMsg) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	➡	Referencia de conexión IMAP
numMsg	Entero largo	➡	Número del mensaje
datosMsg	Texto	➡	Elementos de datos a recuperar
valoresMsg	Texto	⬅	Valores de los datos recuperados
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_MsgFetch* permite al usuario recuperar uno o varios elementos de datos simples de un mensaje especificado sin necesidad de descargar el mensaje.

imap_ID es una referencia entero largo a una conexión abierta creada con *IMAP_Login*.

numMsg es un valor entero largo que indica el mensaje a examinar. Este valor representa la posición de un mensaje dentro de la lista actual de los mensajes. El *numMsg* de un mensaje no es un valor estable, difiere de una sesión a otra.

datosMsg es una variable texto que indica el o los elementos a recuperar. En el caso de varios elementos de datos, un carácter de espacio debe separar cada uno de ellos. Hay dos tipos de elementos de datos:

- Elementos de datos simples, que sólo recuperan una pieza de información y
- Macro elementos de datos, que recuperan varias piezas de información a la vez. Tres macros definen los conjuntos de información actuales disponibles. Una macro debe ser utilizada sola, es decir sin otra macro ni elemento de datos.

Para obtener más información sobre los elementos de datos, consulte los párrafos "Elementos de datos simples" y "Macro elementos de datos"

valoresMsg es una variable texto que puede devolver un par simple Elemento de datos/Valor de datos, o una lista de pares Elemento de datos/Valor de datos.

- En el primer caso, la estructura del texto devuelto es la siguiente: *Elemento de datos* + *Espacio* + *Valor de datos*
- En el segundo caso, la estructura del texto devuelto es la siguiente: *Elemento de datos1* + *Espacio* + *Valor de datos1*+*Elemento de datos2* + *Espacio* + *Valor de datos2.*

valoresMsg puede contener una lista entre paréntesis, una cadena entre comillas o una sola cadena en función del parámetro *datosMsg*.

- La lista entre paréntesis está estructurado de la siguiente manera (ver el ejemplo **FLAGS**): (Valor de datos1 + Espacio +Valor de datos2). Si la lista entre paréntesis devuelve sólo paréntesis, esto significa que no hay ningún valor de datos. Esta regla no se aplica a listas de direcciones entre paréntesis (ver el ejemplo **ENVELOPE**).
- Las cadenas entre comillas se estructuran de la siguiente manera (ver el ejemplo **INTERNALDATE**): Elemento de datos + comillas + valores de datos + Comillas. Cuando un valor de datos es inexistente, se devuelve una cadena vacía "".
- Las cadenas simples (sin comillas) indican los valores de tipo entero, entero largo o numérico y se estructuran de la siguiente manera: Elemento de datos + espacio + Valor de datos. En este caso, lo más probable es que tenga que convertir al tipo adecuado (ver el ejemplo **UID**).

Nota: las comillas se utilizan por lo general cuando las cadenas contienen caracteres especiales, como un espacio o paréntesis. Por lo tanto, al analizar la cadena de caracteres devuelta por el comando **IMAP_Fetch**, los caracteres comillas se tienen en cuenta al procesar el contenido de la cadena.

Elementos de datos simples

- **INTERNALDATE**
  Recupera la fecha y la hora internas en el servidor IMAP. Esta no es la fecha y hora devueltas por

el encabezado "Date", sino la fecha y hora que indican el momento en que el mensaje fue recibido. Para los mensajes enviados vía un servidor SMTP, esta información indica por lo general la fecha y la hora de la entrega final del mensaje. Para los mensajes enviados vía luego de un comando **IMAP_Copy**, esta información indica generalmente la fecha y la hora internas del mensaje fuente.
El elemento de datos **INTERNALDATE** devuelve una cadena entre comillas.

Ejemplo:

```
datosMsg:="INTERNALDATE"
$Err:=IMAP_MsgFetch(imap_ID;1;datosMsg;valoresMsg)
```

valoresMsg devuelve INTERNALDATE "17-Jul-2001 15:45:37 +0200"

- **FLAGS**
  Recupera entre paréntesis la lista de los marcadores definidos para el mensaje. Los marcadores están separados por espacios.

Ejemplo:

```
datosMsg:="FLAGS"
$Err:=IMAP_MsgFetch(imap_ID;1;datosMsg;valoresMsg)
```

valoresMsg devuelve FLAGS () si no hay ningún marcador definido para el mensaje especificado.
valoresMsg devuelve FLAGS (\Seen \Answered) si los marcadores **\Seen** y **\Answered** están definidos para el mensaje.

- **RFC822.SIZE**
  Recupera el número de bytes del mensaje, expresado en el formato RFC-822. El nombre del elemento de datos está separado del valor devuelto por un espacio. Se devuelve una cadena sin comillas, lo que significa que probablemente necesite convertir esta cadena en entero largo (ver el ejemplo **UID**).

Ejemplo:

```
datosMsg:="RFC822.SIZE"
$Err:=IMAP_MsgFetch(imap_ID;1;datosMsg;valoresMsg)
```

valoresMsg devuelve RFC822.SIZE 99599

- **ENVELOPE**
  Recupera entre paréntesis la lista que describe la parte del encabezado del mensaje. El servidor trata esta parte analizando los campos de encabezado y los valores por defecto si es necesario. Los campos de encabezado se devuelven en el siguiente orden: fecha, asunto, de, responder a, para, cc, bcc, en respuesta a y "message-id")
  Ejemplo:

```
datosMsg:="ENVELOPE"
$Err:=IMAP_MsgFetch(imap_ID;1;datosMsg;valoresMsg)
```

valoresMsg devuelve ENVELOPE ("Tue, 17 Jul 2001 17:26:34 +0200" "Test" (("RSmith" NIL "RSmith" "test")) (("RSmith" NIL "RSmith" "test")) (("RSmith" NIL "RSmith" "test")) (("RSmith" NIL "RSmith" "test")) () () "" "<ee6b33a.-1@Mail.x6foadRIbnm>")

Date:	"Tue, 17 Jul 2001 17:26:34 +0200"	Encabezado date
Subject:	"Test"	subject header
From:	(("RSmith" NIL "RSmith" "test"))	Estructura de dirección
Sender:	(("RSmith" NIL "RSmith" "test"))	Estructura de dirección
reply-to:	(("RSmith" NIL "RSmith" "test"))	Estructura de dirección
to:	(("RSmith" NIL "RSmith" "test"))	Estructura de dirección
cc:	()	Encabezado Cc
bcc:	()	Encabezado Bcc
in-reply-to:	""	In-reply-to header
message-id:	"<ee6b33a.-1@Mail.x6foadRIbnm>"	message-id header

Los encabezados de, para, en respuesta a, cc y bcc son listas entre paréntesis de estructuras de direcciones. Una estructura de direcciones es una lista que describe una dirección de correo electrónico. Los campos de una estructura de dirección se presentan en el siguiente orden: nombre, [SMTP] at-domain-list (source route), nombre del buzón y nombre del servidor. Por ejemplo, *(("RSmith" NIL "RSmith" "test"))*.

- **BODY**
  Devuelve la misma información que **BODYSTRUCTURE** excepto para los datos de extensión (ver **BODYSTRUCTURE**).
  Ejemplo:

```
datosMsg:="BODY"
$Err:=IMAP_MsgFetch(imap_ID;1;datosMsg;valoresMsg)
```

valoresMsg devuelve BODY ("TEXT" "PLAIN" ("CHARSET" "us-ascii") NIL NIL "8BIT" 8 1)

- **BODYSTRUCTURE**
  Recupera la estructura MIME del mensaje. El servidor trata esta parte analizando los campos del encabezado MIME en el encabezado y cuerpo del mensaje. Este elemento de datos es particularmente útil para analizar el contenido de un mensaje sin descargarlo. Por ejemplo, puede probar rápidamente el tamaño de cada parte o los nombres de los archivos adjuntos.
  BODYSTRUCTURE devuelveuna lista entre paréntesis, cadenas entre comillas y cadenas sin comillas.
  En función del contenido del mensaje, **BODYSTRUCTURE** devolverá una lista "non- multipart" o una anidada ("multipart"):
  - Lista "**non-multipart**": similar, por ejemplo, a un correo electrónico "non-multipart"; la estructura del cuerpo de un mensaje texto simple de 48 líneas y 2279 bytes puede ser la siguiente: *("TEXT" "PLAIN" ("CHARSET" "us-ascii") NIL NIL "8BIT" 8 1 NIL NIL NIL).*
    Los campos simples de una lista "non-multipart" entre paréntesis aparece en el siguiente orden:

body type	Cadena que da el tipo del contenido de media (Content-type: media type ej. TEXT)
body subtype	Cadena que da el subtipo del contenido de media (Content-type: subtype ej. PLAIN)
body parameter parenthesized list	Lista entre paréntesis de pares atributos/valores [ej. ("CHARSET" "US-ASCII" "NAME" "cc.diff") donde "US-ASCII" es el valor de "CHARSET" y "cc.diff" es el valor de "NAME".
body id	Cadena que da el número de ID del contenido (permite a un cuerpo hacer referencia a otro). Por lo tanto, los cuerpos pueden etiquetarse utilizando el campo de encabezado "Content-ID". El valor Content-ID tiene una sintaxis particular en el caso de un tipo de media multipart/alternative. Ver la explicación en la sección de la RFC 2046 relativa a los casos multipart/alternative.
body description	Cadena que describe el contenido
body encoding	Cadena que da la codificación de transferencia del contenido (Content-Transfer-Encoding)
body size	Valor numérico que indica que el tamaño del cuerpo en bytes. Note que este es el tamaño durante la codificación de transferencia y no el tamaño resultante después de la decodificación.

- Un cuerpo de tipo MESSAGE y subtipo RFC822 contiene, justo después los campos simples, la estructura de sobre, la estructura de cuerpos y el tamaño en líneas de texto del mensaje encapsulado.
- Un cuerpo de tipo TEXT contiene, justo después los campos simples el tamaño en líneas de texto del cuerpo. Note que este es el tamaño durante la codificación y no el tamaño resultante después de la decodificación. Los datos de extensión siguen los campos simples y los campos de tipo listados arriba. Los datos de extensión no se devuelven con el elemento **BODY**, pero pueden devolverse con **BODYSTRUCTURE**.

Si están presentes, los datos de extensión de una lista "non multipart" entre paréntesis deben aparecer en el siguiente orden:

body MD5	Cadena que da el valor MD5 del cuerpo, como se define en [MD5]
body disposition	Lista entre paréntesis que consiste de una cadena de tipo de disposición seguida por una lista entre paréntesis de pares de atributos/valores como se definió en [DISPOSITION]
body language	Cadena o lista entre paréntesis que indica el lenguaje del cuerpo como se definió en [LANGUAGE-TAGS]

Ejemplo: *("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 2279 48 NIL NIL NIL)*
Descripción:*("bodytype" "bodysubtype" (BodyParameterParenthesizedList) bodyId bodyDescription "bodyEncoding" BodySize BodySizeInTextLines ExtensionDataBODYmd5 ExtensionDataBodyDisposition ExtensionDataBodyLanguage)*

- Lista "**multipart**" entre paréntesis: este es el caso del e-mail multipart; incluye una lista "non-multipart" entre paréntesis.

Los paréntesis anidados indican dos partes múltiples (multiple parts). El primer elemento de la lista entre paréntesis es un cuerpo anidado y no un tipo de cuerpo. El segundo elemento de la lista es el subtipo multipart (mixed, digest, parallel, alternative, etc.).
El subtipo multipart está seguido por los datos de extensión. Cuando están presentes, los datos de extensión deben aparecer en el siguiente orden:

body parameter	Lista entre paréntesis de pares atributos/valores
body disposition	Lista entre paréntesis consiste de una cadena de tipo de disposición seguida de una lista entre paréntesis de pares atributos/valores como se definió en [DISPOSITION]
body language	Cadena o lista entre paréntesis que indica el lenguaje del cuerpo como se define en [LANGUAGE-TAGS]

Los datos de extensión posteriores no están aún definidos en esta versión del protocolo. Estos datos de extensión pueden consistir de cero o más NILs, cadenas, números o listas entre paréntesis de estos datos. Las implementaciones clientes utilizan el elemento de datos **BODYSTRUCTURE** DEBEN estar preparadas para aceptar tales datos de extensión. Las implementaciones Server NO DEBEN enviar tales datos de extensión hasta que haya sido definido en una revisión del protocolo.
Ejemplo: *BODYSTRUCTURE (("TEXT" "PLAIN" ("CHARSET" "us-ascii") NIL NIL "7BIT" 22 1 NIL NIL NIL)("APPLICATION" "BYTE-STREAM" ("NAME" "casta37.jpg" "X-MAC-TYPE" "4A504547" "X-MAC-CREATOR" "6F676C65") NIL NIL "BASE64" 98642 NIL ("ATTACHMENT" ("FILENAME" "casta37.jpg")) NIL) "MIXED" ("BOUNDARY" "4D_====================1385356==") NIL NIL)*
Descripción: *(("bodytype" "bodysubtype" (BodyParameterParenthesizedList) bodyId bodyDescription "bodyEncoding" BodySize BodySizeInTextLines ExtensionDataBODYmd5 ExtensionDataBodyDisposition ExtensionDataBodyLanguage) ("bodytype" "bodysubtype" (BodyParameterParenthesizedList) bodyId bodyDescription "bodyEncoding" BodySize BodySizeInTextLines ExtensionDataBODYmd5 ExtensionDataBodyDisposition ExtensionDataBodyLanguage) "multipartSubtype" (ExtensionDataBodyParameterList) ExtensionDataBodyDisposition ExtensionDataBodyLanguage))*

- **UID**
  Recupera el número de identificación único del mensaje, equivalente a ejecutar *IMAP_UIDToMsgNum*. Como este número se devuelve en un área de texto, debe convertirlo en Entero largo.
  Ejemplo:

```
datosMsg:="UID"
$Err:=IMAP_MsgFetch(imap_ID;1;datosMsg;valoresMsg)
```

valoresMsg devuelve UID 250000186

Para obtener un entero largo:

```
C_LONGINT(vLongint)
VLongint:=Num("250000186")
```

Macro elementos de datos

- **FAST**
  Macro equivalente a: (FLAGS INTERNALDATE RFC822.SIZE)
  Ejemplo:

```
$Err:=IMAP_MsgFetch(imap_ID;msgNum;"FAST";msgDataItemValue)
```

valoresMsg devuelve *"FLAGS (\Seen \Answered) INTERNALDATE "17-Jul-2001 15:45:37 +0200" RFC822.SIZE 99599"*

- **ALL**
  Macro equivalente a: *(FLAGS INTERNALDATE RFC822.SIZE ENVELOPE)*
- **FULL**
  Macro equivalente a: *(FLAGS INTERNALDATE RFC822.SIZE ENVELOPE BODY)*

IMAP_MsgInfo

IMAP_MsgInfo (imap_ID ; numMsg ; tamMsg ; IDunico) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	→	Referencia de conexión IMAP
numMsg	Entero largo	→	Número del mensaje
tamMsg	Entero largo	←	Tamaño del mensaje
IDunico	Entero largo	←	ID único de un mensaje en el servidor
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_MsgInfo* devuelve información sobre el mensaje identificado por *numMsg* en el buzón actual. Se devuelve información sobre el tamaño del mensaje y su identificador único.

imap_ID es una referencia entero largo de una conexión abierta creada con *IMAP_Login*.

numMsg es un valor entero largo que indica cuál es el mensaje del buzón del que desea recuperar información. El *numMsg* representa la posición del mensaje en la lista actual de los mensajes. El *numMsg* de un mensaje no es un valor estable, difiere de una sesión a otra en función de las operaciones efectuadas.

tamMsg devuelve el tamaño del mensaje.

IDunico es una variable entero largo que indica el identificador único del mensaje en el servidor. *IDunico* es un valor asignado al mensaje por el servidor IMAP4. Este valor no cambia de una sesión a otra, a diferencia de *numMsg*. El valor uniqueID es una referencia útil para comprobar si la base de datos ya ha descargado un mensaje desde el servidor.

IMAP_MsgLst

IMAP_MsgLst (imap_ID ; primerMsg ; ultimoMsg ; arrayEncabMsg ; arrayNumMsg ; arrayIDMsg ; arrayValoresMsg) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	→	Referencia de conexión IMAP
primerMsg	Entero largo	→	Número del primer mensaje
ultimoMsg	Entero largo	→	Número del último mensaje
arrayEncabMsg	Array cadena	→	Array de encabezados a recuperar
arrayNumMsg	Array entero largo	←	Array de los números de mensajes
arrayIDMsg	Array entero largo	←	Array ID únicos
arrayValoresMsg	Array alfa 2D, Array texto 2D	←	Array 2D de los valores de los encabezados
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_MsgLst* se utiliza para obtener información específica sobre el contenido de los buzones. Permite al usuario solicitar columnas específicas de la lista de mensajes. Este comando sólo puede devolver los valores de los encabezados, no se puede utilizar para recuperar el cuerpo de un mensaje. El contenido de los encabezados es decodificado automáticamente y convertido si es necesario (ver la descripción del comando *POP3_Charset* para obtener más información sobre la decodificación y las reglas de conversión).

imap_ID es una referencia entero largo a una conexión abierta creada con *IMAP_Login*.

primerMsg es un número entero largo que especifica el número del primer mensaje a examinar. El número de mensaje es un valor que representa la posición de un mensaje en la lista de todos los mensajes en el buzón de trabajo actual.

ultimoMsg es un número entero largo que indica el número del último mensaje a examinar. El número de mensaje es un valor que representa la posición de un mensaje en la lista de todos los mensajes en el buzón de trabajo actual.

Nota: los comandos *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* y *IMAP_CopyToMB* no devuelven un error si *primerMsg* es mayor que el *ultimoMsg*. En caso de que esto ocurra, el comando no hace nada.

arrayEncabMsg es un array alfanumérico o texto que lista los encabezados a recuperar.

arrayNumMsg devuelve los números de los mensajes entre *primerMsg* y *ultimoMsg*.

arrayIdMsg recibe los identificadores únicos de los mensajes entre *primerMsg* y *ultimoMsg*.

arrayValoresMsg recibe los datos para cada encabezado especificado por *arrayEncabMsg*. A cada encabezado solicitado le corresponde una línea del array *arrayValoresMsg*.

Ejemplo

```
aHeaders{1}:="Date:"
aHeaders{2}:="From:"
aHeaders{3}:="Subject:"
IMAP_MsgLst(IMAP_ID;vStart;vEnd;aHeaders;aMsgNum;aMsgId;aValues)
```

aValues{1}{1} contiene por ejemplo "Jueves, 19 de noviembre 00:24:02 -0800"
aValues{2}{1} contiene por ejemplo "Jack@4d.com"
aValues{3}{1} contiene por ejemplo "Llame a su esposa"

Los errores se manejan de esta forma:

1) Sólo se devuelven los códigos de error relacionados con la comunicación. Si el comando no puede completar su tarea debido a un error (red, sintaxis, servidor, etc.), se devuelve el código de error correspondiente.

2) Si un mensaje dentro del rango especificado de mensajes no existe o contiene un error:

- No se crea ningún elemento del array para ese mensaje.

- No se devuelve ningún código de error.

3) La imposibilidad de localizar uno o varios encabezados en un mensaje no constituye un error:

- Un elemento del array se crea para el mensaje.
- Los elementos de los arrays *aMsgNum* y *aMsgId* contendrán los valores apropiados.
- Para cada encabezado que no existe en el mensaje, se devolverá una cadena vacía en ese elemento del array.
- No se devuelve ningún código de error.

IMAP_MsgLstInfo

IMAP_MsgLstInfo (imap_ID ; primerMsg ; ultimoMsg ; arrayTamMsg ; arrayNumMsg ; arrayIDMsg) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	→	Referencia de conexión IMAP
primerMsg	Entero largo	→	Número del primer mensaje
ultimoMsg	Entero largo	→	Número del último mensaje
arrayTamMsg	Array entero largo	←	Array de tamaños
arrayNumMsg	Array entero largo	←	Array de los números de mensajes
arrayIDMsg	Array entero largo	←	Array de los ID únicos
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_MsgLstInfo* devuelve información sobre un conjunto de mensajes en el buzón actual (definido por el comando *IMAP_SetCurrentMB*). La información se devuelve en tres arrays, cada elemento de los arrays corresponde a un mensaje. Se devuelve información sobre el tamaño y el número del mensaje. Los arrays pasados como parámetros deben ser declarados previamente, aunque pueden ser de cualquier tamaño. El comando *IMAP_MsgLstInfo* redimensiona cada array al número de mensajes recuperados.

El comando *IMAP_MsgLstInfo* no devuelve un número de error si no puede recuperar la información en un mensaje dentro de la lista de mensajes actual. Si se produce un error, no se crea ningún elemento en los arrays para el mensaje de problema. Si el comando lee todos los mensajes con éxito, *arrayNumMsg* debe contener valores numéricos en un orden secuencial. En caso de problemas, se perderá la secuencia numérica de *arrayNumMsg*.

imap_ID es una referencia entero largo a una conexión abierta creada con *IMAP_Login*.

primerMsg es un número entero largo que especifica el número del primer mensaje a examinar. El número de mensaje es un valor que representa la posición de un mensaje en la lista de todos los mensajes en el buzón de trabajo actual.

ultimoMsg es un número entero largo que indica el número del último mensaje a examinar. El número de mensaje es un valor que representa la posición de un mensaje en la lista de todos los mensajes en el buzón de trabajo actual.

Nota: los comandos *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* y *IMAP_CopyToMB* no devuelven un error si *primerMsg* es mayor que el *ultimoMsg*. En caso de que esto ocurra, el comando no hace nada.

arrayTamMsg recibe el tamaño de cada mensaje entre *primerMsg* y *ultimoMsg*.

arrayNumMsg devuelve los números de los mensajes entre *primerMsg* y *ultimoMsg*.

arrayValoresMsg recibe los identificadores únicos de los mensajes entre *primerMsg* y *ultimoMsg*.

IMAP_MsgNumToUID

IMAP_MsgNumToUID (imap_ID ; numMsg ; IDunico) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	→	Referencia de conexión IMAP
numMsg	Entero largo	→	Número del mensaje
IDunico	Entero largo	←	ID único de un mensaje en el servidor
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_MsgNumToUID* recupera el número de identificación único de un mensaje del buzón actual referenciado por *imap_ID* a partir de su número actual en la lista de mensajes.

imap_ID es una referencia entero largo de una conexión abierta creada con *IMAP_Login*.

numMsg contiene el número actual del mensaje (su posición en la lista de mensajes actuales). Si el *IDunico* no se puede encontrar en el servidor, se devuelve cero en *numMsg* y no se devuelve error.

IDunico es un valor entero largo que devuelve el número de identificación único del mensaje en el servidor IMAP.

IMAP_RenameMB

IMAP_RenameMB (imap_ID ; nomBuzon ; nuevNomBuzon) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	➡	Referencia de conexión IMAP
nomBuzon	Texto	➡	Nombre del buzón a renombrar
nuevNomBuzon	Texto	➡	Nuevo nombre de buzón
resultado	Entero	⮌	Código de error

Descripción

El comando *IMAP_RenameMB* cambia el nombre de un buzón. Se produce un error si intenta cambiar el nombre de un buzón de un nombre de buzón que no existe o si utiliza un nombre de un buzón que ya existe.

Nota: es posible renombrar el buzón INBOX. Se pasan todos los mensajes del buzón INBOX a un nuevo buzón con el nuevo nombre, dejando el buzón INBOX vacío. Si el servidor IMAP permite la creación de sub buzones en el INBOX, estos no se ven afectados por el cambio de nombre.

imap_ID es una referencia entero largo de una conexión abierta creada con *IMAP_Login*.

nomBuzon es el nombre completo del buzón a renombrar (ver reglas de nombres en la introducción IMAP).

nuevNomBuzon contiene el nuevo nombre completo del buzón.

IMAP_Search

IMAP_Search (imap_ID ; critBusq ; arrayNumMsg) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	➡	Referencia de conexión IMAP
critBusq	Texto	➡	Criterio de búsqueda
arrayNumMsg	Array entero largo	⬅	Array de los números de mensajes
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_Search* busca los mensajes que corresponden a los criterios definidos en el buzón actual. El parámetro *criterioBusq* contiene una o más palabras claves de búsqueda. *arrayNumMsg* devuelve la lista de los números de los mensajes encontrados por la búsqueda.

imap_ID contiene la referencia de una sesión abierta con *IMAP_Login*.

criterioBusq contiene una o más palabras claves de búsqueda (ver "Palabras claves de búsqueda" al final de este párrafo) asociado o no a los valores de búsqueda. Un criterio de búsqueda puede ser una palabra clave simple o una lista de palabras claves entre paréntesis. Por ejemplo:

Palabraclave1 = FLAGGED
Palabraclave2 = NOT FLAGGED
Palabraclave3 = FLAGGED DRAFT

Nota: generalmente, la búsqueda no tiene en cuenta las mayúsculas y minúsculas.

- Si el parámetro *criterioBusq* es una cadena vacía, la búsqueda equivaldrá a "seleccionar todos":

```
IMAP_Search(imap_ID;"";arrayNumMsg)
```

  ... devuelve todos los mensajes del buzón actual.

- Si el parámetro *criterioBusq* contiene varias palabras claves de búsqueda, el resultado es la intersección (función Y) de todos los mensajes que contienen esas palabras claves.
  criterioBusq = FLAGGED FROM "SMITH"
  ... devuelve todos los mensajes con el marcador **\Flagged** Y enviados por Smith.

- Puede utilizar los operadores **OR** o **NOT**:
  criterioBusq = OR SEEN FLAGGED
  ... devuelve todos los mensajes con el marcador **\Seen** O **\Flagged**
  criterioBusq = NOT SEEN
  ... devuelve todos los mensajes sin el marcador **\Seen**.
  criterioBusq = HEADER CONTENT-TYPE "MIXED" NOT HEADER CONTENT-TYPE "TEXT"...
  ... devuelve todos los mensajes donde el encabezado content-type contiene el tipo "Mixed" y no contiene "Text".
  criterioBusq = HEADER CONTENT-TYPE "E" NOT SUBJECT "o" NOT HEADER CONTENT-TYPE "MIXED"
  ... devuelve todos los mensajes donde el encabezado content-type contiene el tipo "e" y donde el encabezado Asunto no contiene "o" y donde el encabezado content-type no es "Mixed".
  criterioBusq = OR (ANSWERED SMALLER 400) (HEADER CONTENT-TYPE "E" NOT SUBJECT "o" NOT HEADER CONTENT-TYPE "MIXED")
  ... devuelve todos los mensajes correspondientes a la primera lista de palabras claves entre paréntesis O a la segunda lista.
  criterioBusq = OR ANSWERED SMALLER 400 (HEADER CONTENT-TYPE "E" NOT SUBJECT "o" NOT HEADER CONTENT-TYPE "MIXED")
  ... devuelve todos los mensajes con el marcador **\Answered** O cuyo tamaño es menor a 400 bytes Y cumplan con los criterios definidos en la lista entre paréntesis.

En los dos últimos ejemplos, el resultado obtenido es diferente cuando elimina los paréntesis de la primera lista de palabras claves.

- El parámetro *criterioBusq* puede contener opcionalmente la instrucción *[CHARSET]*. Esta instrucción está compuesta de la palabra "CHARSET" seguida por un conjunto de caracteres definido *[CHARSET] (US ASCII, ISO-8859)*. Esto indica el charset de la cadena *criterioBusq*. Por lo tanto, debe convertir la cadena de búsqueda en el conjunto de caracteres especificado si utiliza la instrucción *[CHARSET]* (ver el comando de 4D **Mac to ISO**).
  Por defecto, 4D Internet Commands codifica la cadena de criterios de búsqueda en "Quotable Printable" si la cadena contiene caracteres extendidos.
  criterioBusq = CHARSET "ISO-8859" BODY "Help"
  ... significa que el criterio de búsqueda utiliza el charset iso-8859 y que el servidor deberá convertir la cadena antes de comenzar la búsqueda, si es necesario.

Tipos de valores de búsqueda

Las palabras claves de búsqueda pueden tratar valores de los siguientes tipos:

- **Valores de tipo fecha**

Los valores de tipo *<fecha>* deben tener este formato: *día+"-"+mes+"-"+año* donde *día* indica la fecha del día en el mes (máx. 2 caracteres), *mes* indica el mes (Ene/Feb/Mar/Abr/May/Jun/Jul/Agos/Sep/Oct/Nov/Dec) y *año* indica el año en 4 caracteres.

Ejemplo: *criterioBusq = SENTBEFORE 1-Feb-2000* (por lo general no es necesario poner entre comillas una fecha ya que no contiene caracteres especiales)

- **Valores de tipo cadena**

Los valores de tipo *<cadena>* pueden contener todo tipo de caracteres y deben estar entre comillas. Si la cadena no contiene caracteres especiales, como espacios por ejemplo, no necesita las comillas. Las comillas le permiten asegurar que la cadena se interpretará correctamente.

Ejemplo: *criterioBusq = FROM "SMITH"*

Nota: las búsquedas basadas en cadenas de caracteres son de tipo "contiene": si el campo de un mensaje contiene al menos la cadena buscada, se encuentra el mensaje. La búsqueda no tiene en cuenta las mayúsculas y minúsculas.

- **Nombres de campos**

Los valores de tipo *<nom de campo>* contienen el nombre de un campo de encabezado.

Ejemplo: *criterioBusq = HEADER CONTENT-TYPE "MIXED"*

- **Marcadores**

Los valores de tipo *<marcador>* aceptan una o varias palabras claves (incluyendo los marcadores estándar), separadas por espacios.

Ejemplo: *criterioBusq = KEYWORD \Flagged \Draft*

- **Conjunto de mensajes**

Los valores de este tipo designan un conjunto de mensajes. Contienen una lista de números de mensajes en un orden ascendente, de 1 al número total de mensajes en el buzón.
Los números están separados por comas; dos puntos indica un intervalo de números.

Ejemplos:
2,4:7,9,12:* representa los mensajes 2,4,5,6,7,9,12,13,14,15 para un buzón con 15 mensajes.

criterioBusq = 1:5 ANSWERED busca entre los mensajes 1 a 5 los que tienen el marcador **\Answered**.

criterioBusq = 2,4 ANSWERED busca entre los mensajes 2 y 4 los mensajes con el marcador **\Answered**.

Palabras de búsqueda

ALL: todos los mensajes en el buzón.

ANSWERED: mensajes con el marcador **\Answered**.

UNANSWERED: mensajes que no tienen el marcador **\Answered**.

DELETED: mensajes con el marcador **\Deleted**.

UNDELETED: mensajes que no tienen el marcador **\Deleted**.

DRAFT: mensajes con el marcador **\Draft**.
UNDRAFT: mensajes que no tienen el marcador **\Draft**.
FLAGGED: mensajes con el marcador **\Flagged**.
UNFLAGGED: mensajes que no tienen el marcador **\Flagged**.
RECENT: mensajes con el marcador **\Recent**.
OLD: mensajes que no tienen el marcador **\Recent**.
SEEN: mensajes con el marcador **\Seen**.
UNSEEN: mensajes que no tienen el marcador **\Seen**.
NEW: mensajes con el marcador **\Recent** y el marcador **\Seen**. Equivale a "(RECENT UNSEEN)".
KEYWORD <marcador>: mensajes con la palabra clave especificada.
UNKEYWORD <marcador>: mensajes que no tienen la palabra clave especificada.
BEFORE <fecha>: mensajes cuya fecha interna es anterior a la fecha especificada.
ON <fecha>: mensajes cuya fecha interna es igual a la fecha especificada.
SINCE <fecha>: mensajes cuya fecha interna es igual o posterior a la fecha especificada.
SENTBEFORE <fecha>: mensajes cuyo encabezado Fecha es anterior a la fecha especificada.
SENTON <fecha>: mensajes cuyo encabezado Fecha es igual a la fecha especificada.
SENTSINCE <fecha>: mensajes cuyo encabezado Fecha es igual o posterior a la fecha especificada.
TO <cadena>: mensajes que contienen la cadena especificada en el encabezado PARA.
FROM <cadena>: mensajes que contienen la cadena especificada en el encabezado DE.
CC <cadena>: mensajes que contienen la cadena especificada en el encabezado CC.
BCC <cadena>: mensajes que contienen la cadena especificada en el encabezado BCC.
SUBJECT <cadena>: mensajes que contienen la cadena especificada en el encabezado Asunto.
BODY <cadena>: mensajes donde el cuerpo contiene la cadena especificada.
TEXT <cadena>: mensajes que contienen la cadena especificada en el encabezado o en el cuerpo.
HEADER <field-name> <cadena>: mensajes donde el encabezado contiene el campo definido y este campo contiene la cadena definida.
UID <ID único del mensaje>: mensajes donde el número único corresponde al valor especificado.
LARGER <n>: mensajes con un tamaño superior al tamaño especificado.
SMALLER <n>: mensajes con un tamaño en bytes inferior al tamaño especificado.
NOT <criterio>: mensajes que no corresponden al criterio especificado.
OR <criterio1> <criterio2>: mensajes que corresponden al primer o segundo criterio especificado.

IMAP_SetCurrentMB

IMAP_SetCurrentMB (imap_ID ; nomBuzon ; numMsg ; numNuevMsg ; listMarc ; marcPermanent ; unicoIDB) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	→	Referencia de conexión IMAP
nomBuzon	Texto	→	Nombre del buzón a seleccionar
numMsg	Entero largo	←	Número de mensajes en el buzón especificado
numNuevMsg	Entero largo	←	Número de mensajes con el marcador \Recent
listMarc	Texto	←	Lista de marcadores utilizados actualmente por el buzón
marcPermanent	Texto	←	Lista de marcadores modificables
unicoIDB	Entero largo	←	Número de identificación único del buzón especificado
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_SetCurrentMB* permite abrir una sesión (es decir, seleccionar el buzón actual) para administrar los mensajes del buzón especificado.

Una sola sesión puede abrirse a la vez durante una conexión, el acceso simultáneo a varios buzones requiere múltiples conexiones (múltiples *IMAP_Login*). El comando *IMAP_SetCurrentMB* cierra automáticamente la sesión actual antes de realizar la nueva selección. Por lo tanto, si se define un buzón como actual y la ejecución del comando *IMAP_SetCurrentMB* falla, no habrá buzón actual.

Puede cerrar una sesión, (es decir, cerrar el buzón actual) sin seleccionar uno nuevo, ejecutando el comando *IMAP_SetCurrentMB* con un *nomBuzon* inexistente y durante la gestión de errores, ejecutando *IMAP_CloseCurrentMB* o *IMAP_Logout*.

imap_ID es una referencia entero largo a una conexión abierta creada con *IMAP_Login*.

nomBuzon es el nombre completo de un buzón a definir como buzón actual.

numMsg devuelve el número de mensajes en el buzón actual (devuelve -1 en caso de error).

numNuevMsg devuelve el número de mensajes recientes en el buzón actual (devuelve -1 en caso de error).

listMarc devuelve la lista completa de marcadores utilizados en el buzón actual. Tenga en cuenta que sólo los marcadores listados en el parámetro *marcPermanent* pueden modificarse.

marcPermanent devuelve la lista de marcadores que pueden modificarse de forma permanente (con excepción del marcador **\Recent**, que es administrado por el servidor IMAP). Note que la cadena *marcPermanent* también puede incluir el marcador especial *, lo que significa que las palabras claves pueden crearse guardando los marcadores en el buzón (ver *IMAP_SetFlags*).

Si *permanentFlags* devuelve una cadena vacía, esto significa que todos los marcadores listados en el parámetro *listMarc* se pueden modificar.

unicoIDB devuelve un identificador único del buzón actual.
Este identificador puede ser particularmente útil si se elimina un buzón y un nuevo buzón con el mismo nombre se crea en una fecha posterior. Dado que el nombre es el mismo, sólo el número único permite al cliente identificar el nuevo buzón.

IMAP_SetFlags

IMAP_SetFlags (imap_ID ; primerMsg ; ultimoMsg ; listMarcMsg ; eliminarOpcion) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	→	Referencia de conexión IMAP
primerMsg	Entero largo	→	Número del primer mensaje
ultimoMsg	Entero largo	→	Número del último mensaje
listMarcMsg	Cadena	→	Marcadores a añadir o eliminar
eliminarOpcion	Entero	→	1 = añadir marcador, 0 = eliminar marcador
resultado	Entero	←	Código de error

Descripción

El comando *IMAP_SetFlags* permite añadir o borrar en una operación varios marcadores asociados a los mensajes del intervalo definido.

El protocolo IMAP permite asociar una lista de marcadores a un mensaje. Hay dos tipos de marcadores: los marcadores **permanentes** y los marcadores de **sesión**.
Los marcadores permanentes se agregan o eliminan permanentemente de los marcadores de los mensajes (ver *IMAP_SetCurrentMB*); es decir, toda modificación de un marcador permanente se conservará durante las sesiones posteriores.
Los cambios efectuados en los marcadores de sesión son válidos sólo para esa sesión.

Los marcadores sistema definidos actualmente son:

- **Seen**: el mensaje ha sido leído.
- **Answered**: el mensaje ha sido respondido.
- **Flagged**: el mensaje está "marcado" como urgente/ de atención especial.
- **Deleted**: el mensaje se borrará con *IMAP_Delete*, *IMAP_CloseCurrentMB*, *IMAP_SetCurrentMB* o *IMAP_Logout*.
- **Draft**: el mensaje está en borrador, es decir, no está completo.
- **Recent**: el mensaje llegó recientemente a este buzón. Este marcador sólo aparece en una sesión; las sesiones posteriores no verán el marcador **\Recent** en este mensaje. Este marcador permanente es administrado por el servidor IMAP y no puede ser modificado por un cliente IMAP utilizando por ejemplo el comando *IMAP_SetFlags*.

Un servidor IMAP puede permitir a un cliente definir marcadores personalizados. En este caso, los marcadores adicionales se llaman palabras claves (keywords) y no comienzan por el carácter "\" (ver el comando IMAP_SetCurrentMB). Esto depende de la implementación del servidor IMAP. En este caso, los marcadores adicionales se llaman palabras claves (keywords) y no comienzan por el carácter "\" (ver el comando *IMAP_SetCurrentMB*).

Nota: si se establece el marcador **\Deleted** y cierra la sesión actual ejecutando *IMAP_SetCurrentMB*, *IMAP_CloseCurrentMB*, *IMAP_Delete* o *IMAP_Logout*, el mensaje se borrará de manera permanente.

imap_ID es una referencia entero largo a una conexión abierta creada con *IMAP_Login*.

primerMsg es un número entero largo que especifica el número del primer mensaje a examinar. El número de mensaje es un valor que representa la posición de un mensaje en la lista de todos los mensajes en el buzón de trabajo actual.

ultimoMsg es un número entero largo que indica el número del último mensaje a examinar. El número de mensaje es un valor que representa la posición de un mensaje en la lista de todos los mensajes en el buzón de trabajo actual.

Nota: los comandos *IMAP_Delete*, *IMAP_MsgLstInfo*, *IMAP_MsgLst*, *IMAP_SetFlags*, *IMAP_GetFlags* y *IMAP_CopyToMB* no devuelven un error si *primerMsg* es mayor que el *ultimoMsg*. En caso de que esto ocurra, el comando no hace nada.

listMarcMsg puede contener uno o varios marcadores. En el caso de varios marcadores, la cadena debe ser una lista de marcadores, separados por espacios. Ver los ejemplos a continuación.

Sólo los marcadores permanentes indicados en la lista *marcPermanentes* pueden ser modificados, (ver *IMAP_SetCurrentMB*).

eliminarOpción es un valor entero que especifica si se debe eliminar o agregar el o los marcador(es) definido(s) por el parámetro *listMarcMsg*:

- Un valor de cero añade los marcadores especificados en *listMarcMsg*.
- Un valor de 1 elimina los marcadores especificados en *listMarcMsg*.

Ejemplo 1

Definición de los marcadores **\Answered** y **\Draft** para los mensajes especificados entre *primerMsg* y *ultimoMsg*:

```
msgFlagsName:="\Answered \Draft"
  ` \Answered y \Draft están separados por un espacio (código ASCII)
IMAP_SetFlags(imap_ID;primerMsg;ultimoMsg;msgFlagsName;0)
```

Ejemplo 2

Eliminación del marcador **\Deleted** para los mensajes entre *inicioMsg* y *ultimoMsg*, cualquiera que sea el estado del marcador:

```
msgFlagsName:="\Deleted"
IMAP_SetFlags(imap_ID;startMsg;endMsg;msgFlagsName;1)
```

Ejemplo 3

Definición del marcador **\Deleted** para los mensajes entre *primerMsg* y *ultimoMsg*, sin importar si el marcador fue definido previamente o no:

```
msgFlagsName:="\Deleted"
IMAP_SetFlags(imap_ID;startMsg;endMsg;msgFlagsName;0)
IMAP_CloseCurrentMB(imap_ID)
  `Cierre del buzón actual y eliminación definitiva de los mensajes especificados.
```

Ejemplo 4

Definición del marcador **\Answered** en función del valor de CheckBoxAnswered:

```
$Error:=IMAP_SetFlags(vImap_ID;$msgNum;$msgNum;"\Answered";Num(CheckBoxAnswered=0))
```

IMAP_SetPrefs

IMAP_SetPrefs (retornoLinea ; carpetaMsg) -> resultado

Parámetro	Tipo		Descripción
retornoLinea	Entero	➡	0 = No eliminar los retornos de línea, 1 = Retirar los retornos de línea, -1 = No cambiar
carpetaMsg	Texto	➡	Ruta de acceso a la carpeta de mensajes (“” = no cambiar)
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_SetPrefs* define las preferencias para todos los comandos.

retornoLinea es un valor entero que especifica como tratar los caracteres de retorno de línea en los mensajes guardados. La mayoría de los servidores IMAP combinan un carácter retorno de carro y un carácter retorno de línea para indicar el final de una línea. Las aplicaciones de Macintosh sólo requieren de un retorno de carro. Esta opción permite a los usuarios retirar los caracteres retorno de línea del texto de los mensajes. Un valor de cero deja los mensajes recuperados en el formato en el que se almacenaron en el servidor IMAP. Un valor de uno retira los caracteres de salto de línea de los mensajes recuperados. Un valor de -1 dejará esta preferencia, como se estableció previamente. La opción por defecto está en 1 y automáticamente retira los saltos de línea que se encuentran en los mensajes.

carpetaMsg indica la ruta de acceso local de la carpeta en la que los mensajes recuperados con el comando *IMAP_Download* se guardan por defecto.

IMAP_SubscribeMB

IMAP_SubscribeMB (imap_ID ; nomBuzon ; inscribBuzon) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	➡	Referencia de conexión IMAP
nomBuzon	Texto	➡	Nombre del buzón a borrar a inscribir o dar de baja
inscribBuzon	Entero	➡	0 = No suscribir; 1= Inscribir
resultado	Entero	⮌	Código de error

Descripción

El comando *IMAP_SubscribeMB* permite modificar la lista de buzones del servidor al cual el usuario está inscrito.

Como tal, el usuario puede optar por reducir una larga lista de buzones disponibles mediante la inscripción a aquellos que quiere ver con frecuencia. Para ello, simplemente tiene que usar el comando *IMAP_ListMBs* pasando 1 en el parámetro *inscribBuzon* (ver *IMAP_ListMBs*).

imap_ID es una referencia entero largo de una conexión abierta creada con *IMAP_Login*.

nomBuzon es el nombre completo del buzón inscribir o dar de baja.

Pase 0 en el parámetro *inscribBuzon* para provocar la baja del usuario; pase 1 para inscribirlo.

IMAP_UIDToMsgNum

IMAP_UIDToMsgNum (imap_ID ; IDunico ; numMsg) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	➡	Referencia de conexión IMAP
IDunico	Entero largo	➡	ID único de un mensaje en el servidor
numMsg	Entero largo	⬅	Número de mensaje
resultado	Entero	↩	Código de error

Descripción

El comando *IMAP_UIDToMsgNum* devuelve el número del mensaje *numMsg* correspondiente al mensaje designado por *IDunico* en el buzón referenciado por *imap_ID*. El *numMsg* de un mensaje es un valor fluctuante con respecto a los otros mensajes de la lista, este comando devuelve la posición actual de un mensaje cuya información pudo haber sido recuperada durante una sesión IMAP anterior.

imap_ID es una referencia entero largo de una conexión abierta creada con *IMAP_Login*.

IDunico es un valor entero largo que indica el identificador único del mensaje a localizar en el servidor IMAP.

numMsg contiene el número actual del mensaje (su posición en la lista de mensajes actuales). Si el *IDunico* no se puede encontrar en el servidor, se devuelve cero en *numMsg* y no se devuelve error.

IMAP_VerifyID

IMAP_VerifyID (imap_ID) -> resultado

Parámetro	Tipo		Descripción
imap_ID	Entero largo	➡	Referencia de conexión IMAP
		⬅	0 = Conexión cerrada
resultado	Entero	↩	Código de error

Descripción

Un servidor IMAP desconecta automáticamente una conexión que no muestra actividad en un período de tiempo determinado por el administrador. Cada comando que interactúa con el servidor IMAP provoca la reinicialización del contador de inactividad. El comando *IMAP_VerifyID* reinicia en cero el contador de inactividad de la conexión IMAP especificada, sin realizar ninguna otra acción. Esto permite al usuario conservar una conexión sesión activa si existe la posibilidad de que la conexión supere el tiempo de espera.

Cuando se ejecuta, el comando *IMAP_VerifyID* verifica que la conexión no se haya cerrado. Si la conexión sigue abierta, el comando le indica al servidor IMAP que reinicie el contador de inactividad para esta conexión. Si la conexión ya ha cerrado, *IMAP_VerifyID* devuelve el error correspondiente y libera la memoria usada por la conexión IMAP y devuelve cero en el parámetro *imap_ID*.

imap_ID es una referencia entero largo a una conexión abierta creada con IMAP_Login.

IC Internet

- Comandos especiales Internet, Presentación
- NET_AddrToName
- NET_Finger
- NET_NameToAddr
- NET_Ping
- NET_Resolve
- NET_Time

Comandos especiales Internet, Presentación

Los comandos de este tema permiten efectuar tareas comunes en Internet. Utilizando estos comandos puede hacer 'Ping' y 'Finger' a una máquina, obtener la hora de un servidor de hora, decodificar un nombre de dominio o una dirección IP, o incluso convertirlos en valores entero largo. Estos comandos se utilizan generalmente en combinación con 4D Internet Commands.

NET_AddrToName

NET_AddrToName (ip_EnteroLargo ; nomServidor ; direccion_IP) -> resultado

Parámetro	Tipo		Descripción
ip_EnteroLargo	Entero largo	→	Referencia a la dirección
nomServidor	Cadena	←	Nombre del servidor
direccion_IP	Cadena	←	Dirección IP
resultado	Entero	↩	Código de error

Descripción

El comando *NET_AddrToName* devuelve el nombre y la dirección IP de un servidor a partir de su referencia única *ip_EnteroLargo*, generada por el comando *NET_NameToAddr*.

Importante: con el fin de asegurar la compatibilidad futura de sus aplicaciones con IPV6, no se recomienda el uso de este comando. Es preferible utilizar *NET_Resolve* que trabaja con cadenas de caracteres.

ip_EnteroLargo indica la referencia única de una dirección IP.

nomServidor devuelve el nombre del servidor en forma de cadena de caracteres.

direccion_IP devuelve la dirección IP del servidor en forma de cadena de caracteres.

Si el nombre del servidor no puede convertirse, *direccion_IP* devolverá una cadena vacía y no devolverá ningún error.

NET_Finger

NET_Finger (nomServidor ; textBusq ; resultados) -> resultado

Parámetro	Tipo		Descripción
nomServidor	Cadena	➡	Nombre o dirección IP del servidor
textBusq	Cadena	➡	Texto de la búsqueda
resultados	Texto	⬅	Resultados de la búsqueda
resultado	Entero	↩	Código de error

Descripción

El comando *NET_Finger* permite obtener información sobre una cuenta de usuario registrada en un servidor. El comando Unix Finger está diseñado para devolver la hora de la última entrada de un usuario, así como información adicional que el usuario elige que presente en sus archivos ".plan" y ".project".

Una búsqueda Finger puede efectuarse de dos formas. Directamente en la máquina del usuario. Por ejemplo, para obtener información sobre "johnt" en "4d.com", puede escribir:

```
$error:=NET_Finger("www.4d.com";"johnt";$fingertext)
```

La misma búsqueda Finger también puede realizarse indirectamente. Una búsqueda indirecta hace la petición a un servidor remoto (que soporta el comando Finger) para realizar la búsqueda. Por ejemplo, la siguiente petición solicita a la máquina identificada por el nombre de dominio "4d.com" realizar una búsqueda Finger del usuario "johnt@4d.com".

```
$error:=NET_Finger("www.4d.com";"johnt@4d.com";$fingertext)
```

Si bien la información principal devuelta en cada caso debe ser la misma, es probable que haya algunas diferencias sutiles en el texto devuelto. Diferentes máquinas pueden tener diferentes opciones de configuración cuando ejecutan el comando Finger y los resultados podrían variar ligeramente. Además, es probable que haya alguna diferencia entre el formato de los resultados de un comando Finger directo e indirecto, las búsquedas indirectas contienen a menudo saltos de línea adicionales.

nomServidor contiene el nombre o la dirección IP del servidor en el cual el usuario identificado por *textBusq* tiene una cuenta.

El parámetro *textBusq* contiene el texto a buscar en el servidor Finger, o un nombre de máquina o dirección IP. Si *textBusq* es un nombre de usuario, el comando buscará en el directorio de los nombres de usuario del servidor. Si *textBusq* es un nombre de máquina o una dirección IP, el comando envía una petición Finger a la máquina especificada por medio del servidor Finger en *nomServidor*.

resultados devuelve el resultado de la búsqueda.

NET_NameToAddr

NET_NameToAddr (nomServidor ; ip_EnteroLargo) -> resultado

Parámetro	Tipo		Descripción
nomServidor	Cadena	→	Nombre o dirección IP del servidor
ip_EnteroLargo	Entero largo	←	Referencia a la dirección
resultado	Entero	↩	Código de error

Descripción

El comando *NET_NameToAddr* devuelve un entero largo que referencia de manera única el nombre o la dirección IP de un servidor.

Importante: con el fin de asegurar en el futuro la compatibilidad de sus aplicaciones con IPV6, el uso de este comando no es recomendable. Es mejor utilizar *NET_Resolve* que trabaja con de cadenas de caracteres.

El parámetro *nomServidor* contiene el nombre o la dirección IP del servidor.

El parámetro *ip_EnteroLargo* devuelve un valor que identifica la dirección IP especificada en el parámetro *nomServidor*. Toda cadena de dirección IP puede convertirse en entero largo.

El comando **NET_NameToAddr** es útil para ahorrar espacio al almacenar datos. Los valores almacenados en el formato Entero largo son más compactos que las cadenas de caracteres.

NET_Ping

NET_Ping (nomServidor ; texto ; activo ; timeout) -> resultado

Parámetro	Tipo		Descripción
nomServidor	Cadena	→	Nombre o dirección IP del servidor
texto	Texto	→	Texto a enviar en el "ping"
activo	Entero	←	1 = Activo, 0 = Timeout/Inactivo
timeout	Entero	→	Número de segundos de espera, 0 = utilizar el valor IT_SetTimeOut
resultado	Entero	↩	Código de error

Descripción

El comando *NET_Ping* ofrece un mecanismo para verificar que una dirección IP remota está activa. Si la máquina de hacer ping está ejecutando el protocolo TCP/IP y la red entre los dos sitios es funcional, se devuelve el estado 'Activo'. Por lo general, la máquina ping no ofrece ninguna indicación a su usuario de la actividad, asumiendo que la máquina puede responder a las peticiones ping (ICMP Echo).

NET_Ping hará ping a un equipo especificado por un nombre de servidor o la dirección IP. Toda máquina con una dirección IP accesible a través de la red puede recibir un ping. Esto incluye las máquinas de los usuarios finales. (Algunos sistemas de seguridad por ejemplo firewalls pueden obstaculizar el ping de las máquinas bajo su protección.)

nomServidor es el nombre del servidor o la dirección IP de la máquina a la cual se le debe hacer ping.

texto es el texto a enviar en el ping. El parámetro texto sólo existe para dar tamaño al paquete TCP enviado durante la ejecución del comando.

activo devuelve un entero que corresponde al estado de la máquina de hacer ping. Un valor de 1 indica que el equipo está activo. Un valor de cero indica que el equipo está inactivo o el comando superó el tiempo de espera antes de recibir una respuesta.

timeout especifica el número de segundos que este comando esperará para que termine el ping. Este es un parámetro opcional, si no se pasa, su valor por defecto es cero. Un valor de cero en este parámetro hará que se utilice el valor de timeout especificado por el comando *IT_SetTimeOut*.

Nota: este parámetro es ignorado en Windows 95/98 y Millenium.

NET_Resolve

NET_Resolve (nomServidor ; ipOServidor) -> resultado

Parámetro	Tipo		Descripción
nomServidor	Cadena	➡	Nombre o dirección IP del servidor
ipOServidor	Cadena	⬅	Devuelve el valor opuesto
resultado	Entero	↩	Código de error

Descripción

El comando *NET_Resolve* devuelve el nombre o la dirección IP de un servidor.

- Si pasa un nombre de servidor en el parámetro *nomServidor*, el comando *NET_Resolve* reenvía la dirección IP en el parámetro *ipOServidor*.
- Si pasa una dirección IP en el parámetro *nomServidor*, el comando *NET_Resolve* reenvía el nombre del servidor registrado por esa máquina en el parámetro *ipOServidor*.

Ejemplo

El siguiente ejemplo pasa primero un nombre de servidor "www.netcom.com" al comando **NET_Resolve** con el fin de obtener su dirección IP. El método hace otra llamada al comando, pasando la dirección IP para obtener su nombre de servidor registrado.

```
C_BOOLEAN($ERR)
C_TEXT($Resolved)  //Puede ser una cadena de caracteres o un valor texto de cualquier tamaño
$ERR:=ERRCHECK("NET_Resolve";NET_Resolve("www.netcom.com";$Resolved))
 //$Resolved reenvío el valor '192.100.81.100'
$ERR:=ERRCHECK("NET_Resolve";NET_Resolve($Resolved;$Resolved))
 //$Resolved reenvío el valor 'www.netcom.com'
```

NET_Time

NET_Time (nomServidor ; fecha ; tiempo ; offset) -> resultado

Parámetro	Tipo		Descripción
nomServidor	Cadena	➡	Nombre o dirección IP del servidor
fecha	Fecha	⬅	Fecha
tiempo	Entero largo	⬅	Hora, expresada en segundos a partir de la media noche
offset	Entero	➡	Horas de desplazamiento
resultado	Entero	↩	Código de error

Descripción

El comando *NET_Time* permite recuperar la fecha y la hora actual de la máquina y aplicar el offset necesario para la conversión a la hora local del usuario.

Nota: este comando no afecta el reloj interno del ordenador.

nomServidor es el nombre o la dirección IP de un servidor NTP (Network Time Protocol).

fecha devuelve la fecha (en formato 4D), después de aplicado el *offset*.

hora devuelve la hora después de aplicado el *offset*. Este valor representa los segundos desde la medianoche en la *fecha*. Ver el ejemplo a continuación de un método de conversión de este valor a una variable hora 4D.

offset es el número de horas a sumar o restar de los valores recibidos. Los servidores de tiempo de Internet expresan sus valores en hora universal (Greenwich Mean Time). Incluso si el servidor de tiempo está en su región geográfica, es probable que deba suministrar un valor de offset para compensar la diferencia entre su hora local y la hora universal.

Ejemplo

El siguiente ejemplo recupera la fecha y la hora del servidor de tiempo en el sitio "apple.com". El comando luego resta las siete horas especificadas en el Offset y devuelve la fecha y la hora resultantes (la hora se expresa como un valor entero largo, que puede convertirse utilizando el comando 4D **Time string**, como se ve a continuación).

```
C_DATE(vNetDate)
C_LONGINT(vNetTime)
C_TIME(vTime)
C_LONGINT(vOffset)
If(ERRCHECK("NET_Time";NET_Time("www.apple.com";vNetDate;vNetTime;-7)))
   vTime:=Time(Time string(vNetTime))  `Convierte la hora entero largo en hora 4D
End if
```

IC POP3 Review Mail

- Recepción de correo, Presentación
- POP3_BoxInfo
- POP3_Charset
- POP3_Delete
- POP3_DownLoad
- POP3_GetMessage
- POP3_GetPrefs
- POP3_Login
- POP3_Logout
- POP3_MsgInfo
- POP3_MsgLst
- POP3_MsgLstInfo
- POP3_Reset
- POP3_SetPrefs
- POP3_UIDToNum
- POP3_VerifyID

Recepción de correo, Presentación

Los comandos POP3 permiten a su base de datos recuperar mensajes de un servidor de correo POP3. Los comandos Internet de 4D son compatibles con MIME y pueden reconocer y extraer los mensajes con varios adjuntos.

Los comandos POP3 se dividen en dos secciones, "IC POP3 Review Mail" y "IC Downloaded Mail", que corresponden a los dos modos de lectura del correo electrónico. Al leer el correo desde un servidor POP3, los mensajes (o información sobre los mensajes) pueden importarse en las estructuras 4D (variables, campos, arrays) o descargarse en el disco. Esta sección describe las posibilidades ofrecidas por 4D Internet Commands para leer los mensajes desde un servidor POP3 en 4D.

El tamaño de los archivos a descargar determina el uso de un modo u otro. Por ejemplo, un solo correo electrónico que contiene un archivo adjunto de 5 MB puede fácilmente superar la capacidad de almacenamiento de la base de datos. La única estructura 4D capaz de almacenar este tamaño es una imagen o un campo BLOB, pero la conversión de un mensaje o de un archivo adjunto a este formato suele ser ineficaz ya que la mensajería cliente debe utilizar grandes recursos de memoria para acceder a la imagen o al BLOB. Para resolver este problema, esta sección tiene un comando *POP3_Download* que trae un mensaje del servidor POP3 al disco local del usuario. Una vez importado al disco, la sección "IC Downloaded Mail" detalla los comandos que se utilizan para manipular archivos locales.

Para el uso de los comandos POP3, es importante entender completamente los parámetros numMsg e IDunico. numMsg es el número de un mensaje en el buzón en el momento que se ejecuta el comando *POP3_Login*. Una vez se realiza la conexión, los mensajes en el buzón se numeran del 1 hasta el número total de elementos en el buzón. Los números se asignan en función del orden en que los mensajes fueron recibidos en el buzón, siendo 1 el más antiguo. Los números asignados a los mensajes sólo son válidos durante el periodo entre *POP3_Login* y *POP3_Logout*.

En el momento de la ejecución de *POP3_Logout*, todo mensaje marcado para eliminación será eliminado. Cuando el usuario inicia sesión de nuevo en el servidor, los mensajes presentes en el buzón se numeran de nuevo del 1 a X. Por ejemplo, si hay 10 mensajes en el buzón de correo y los mensajes numerados del 1 al 5 se eliminan, los mensajes del 6 al 10 pasarán a ser del 1 al 5, la próxima vez que el usuario inicie sesión en el buzón.

Para ilustrar este funcionamiento, supongamos que se conecta a un servidor POP3 y obtiene la siguiente lista de mensajes:

#	IDunico	Fecha	De	Asunto
1	bd573a4dbd573a4d	1 Jul 1998 …	jimw@acme.com	Clientes potenciales...
2	bd574dc7bd574dc7	1 Jul 1998 …	frank@acme.com	Orden de licencia en sitio
3	bd575f06bd575f06	3 Jul 1998 …	joe@acme.com	Alguien quiere ir a almorzar?
4	bd5761d4bd5761d4	4 Jul 1998 …	kelly@acme.com	Su esposa llamó...
5	bd577dc7db577dc5	4 Jul 1995 …	track@fedex.com	Seguimiento FedEx

Durante la sesión elimina los mensajes 3 y 4. Al cerrar la sesión sus solicitudes de eliminación se ejecutan. Cuando vuelva al servidor, la lista de mensajes se reenumerará así:

#	IDunico	Fecha	De	Asunto
1	bd573a4dbd573a4d	1 Jul 1998 …	jimw@acme.com	Clientes potenciales...
2	bd574dc7bd574dc7	1 Jul 1998 …	frank@acme.com	Orden de licencia en sitio
3	bd577dc7db577dc5	4 Jul 1995 …	track@fedex.com	Seguimiento FedEx[

numMsg no es un valor estático con relación a un mensaje específico y cambiará de una sesión a otra depende de su relación con otros mensajes en el buzón en el momento de la apertura de la sesión. El *IDunico* sin embargo es un número único asignado al mensaje cuando es recibido por el servidor. Este número se calcula con la hora y la fecha en las que se recibe el mensaje y es un valor asignado por el servidor POP3. Desafortunadamente, los servidores POP3 no utilizan el IDunico como referencia principal para sus mensajes. Al utilizar los comandos POP3 tendrá que especificar el *numMsg* como referencia para los mensajes en el servidor. Los desarrolladores deben tener cuidado al desarrollar soluciones que referencian los mensajes en la base de datos, dejando el cuerpo del mensaje en el servidor.

Nota: para mayor flexibilidad, los comandos Internet de 4D permiten pasar directamente una referencia de conexión POP3, IMAP o FTP a los comandos de bajo nivel TCP y viceversa. Para más información, consulte la sección **Rutinas de bajo nivel, Presentación**.

POP3_BoxInfo

POP3_BoxInfo (pop3_ID ; numMsg ; tamMsg) -> resultado

Parámetro	Tipo		Descripción
pop3_ID	Entero largo	➡	Referencia de una conexión POP3
numMsg	Entero largo	⬅	Número de mensajes
tamMsg	Entero largo	⬅	Tamaño de los mensajes
resultado	Entero	↩	Código de error

Descripción

El comando *POP3_BoxInfo* devuelve el número y el tamaño de los mensajes presentes en el buzón de la sesión referenciada por *pop3_ID*.

pop3_ID contiene la referencia de una sesión abierta con *POP3_Login*.

numMsg devuelve el número de mensajes presentes en el buzón.

tamMsg devuelve el tamaño total de los mensajes presentes en el buzón.

POP3_Charset

POP3_Charset (decodEncab ; conjCuerpos) -> resultado

Parámetro	Tipo		Descripción
decodEncab	Entero	➡	-1 = Utilizar el parámetro actual, 0 = No hace nada, 1 = Convertir en el conjunto de caracteres Mac OS si ISO-8859-1 o ISO-2022-JP, decodificar los caracteres extendidos
conjCuerpos	Entero	➡	-1 = Utilizar el parámetro actual, 0 = No hace nada, 1 = Convertir en el conjunto de caracteres Mac OS si ISO-8859-1 o ISO-2022-JP
resultado	Entero	↩	Código de error

Descripción

El comando *POP3_Charset* automatiza el tratamiento de los caracteres extendidos en los mensajes mientras los procesa con los comandos POP3 y MSG. Si este comando no se llama o tiene los parámetros en 0, los Internet Commands versión 6.7 o superior funcionarán del mismo modo que la versión 6.5.x.

POP3_Charset permite, en primer lugar, definir si los encabezados con caracteres extendidos deben ser decodificados y en segundo lugar, si debe convertirse el conjunto de caracteres utilizado en el cuerpo del mensaje y en los encabezados.
Este comando es especialmente útil para el tratamiento de caracteres extendidos incluidos en los encabezados tales como el "Asunto" o direcciones de correo electrónico (por ejemplo, para decodificar una dirección como "*=?ISO-8859-1?Q?Test=E9?= <test@n.net >*").

El parámetro *decodEncab* define los tratamientos a aplicar a los campos de encabezado durante la ejecución de los comandos *POP3_MsgLst* o *MSG_FindHeader*(ver Nota de compatibilidad). El valor por defecto es 0.

- -1: Usar la configuración actual;
- 0: No hacer nada;
- 1: Los encabezados son decodificados si es necesario. Si el encabezado es decodificado y si el conjunto de caracteres especificado es ISO-8859-1 o ISO-2022-JP, los encabezados se convierten utilizando ASCII Mac OS código o Shift-JIS, respectivamente.

Nota de compatibilidad (versión 6.8.1): *POP3_Charset* se aplica al comando *MSG_FindHeader* únicamente si el comando *MSG_Charset* no se ha ejecutado con anterioridad.

El parámetro *conjCuerpos* define el tratamiento a aplicar al cuerpo del mensaje durante la ejecución del comando *MSG_GetBody* (ver Nota de compatibilidad). El valor por defecto es 0.

- -1: Usar la configuración actual;
- 0: No hacer nada;
- 1: Si el conjunto de caracteres especificado en el campo "Body-Content-Type" es ISO-8859-1 o ISO-2022-JP, el texto del cuerpo del mensaje se convierte utilizando ASCII Mac OS o Shift-JIS, respectivamente.

Nota de compatibilidad (versión 6.8.1): *POP3_Charset* se aplica al comando *MSG_GetBody* únicamente si el comando *MSG_Charset* no se ha ejecutado con anterioridad.

Ejemplo 1

Con 4D Internet Commands versión 6.5.x:

```
$Err:=MSG_FindHeader($msgfile;"From";$from)
$from:=ISO to Mac($from)
$Err:=MSG_FindHeader($msgfile;"To";$to)
$to:=ISO to Mac($to)
$Err:=MSG_FindHeader($msgfile;"Cc";$cc)
$cc:=ISO to Mac($cc)
$Err:=MSG_FindHeader($msgfile;"Subject";$subject)
$subject:=ISO to Mac($subject)
```

```
$Err:=MSG_MessageSize($msgfile;$HdrSize;$BdySize;$MsgSize)
$Err:=MSG_GetBody($msgfile;0;$BdySize;$BodyContent)
$BodyContent:=ISO to Mac($BodyContent)
```

Ejemplo 2

Con Internet Commands versión 6.7.x:

```
$Err:=POP3_Charset(1;1)
$Err:=MSG_FindHeader($msgfile;"From";$from)
$Err:=MSG_FindHeader($msgfile;"To";$to)
$Err:=MSG_FindHeader($msgfile;"Cc";$cc)
$Err:=MSG_FindHeader($msgfile;"Subject";$subject)

$Err:=MSG_MessageSize($msgfile;$HdrSize;$BdySize;$MsgSize)
$Err:=MSG_GetBody($msgfile;0;$BdySize;$BodyContent)
```

Ejemplo 3

Con Internet Commands versión 6.8.x:

```
$Err:=MSG_Charset(1;1)
$Err:=MSG_FindHeader($msgfile;"From";$from)
$Err:=MSG_FindHeader($msgfile;"To";$to)
$Err:=MSG_FindHeader($msgfile;"Cc";$cc)
$Err:=MSG_FindHeader($msgfile;"Subject";$subject)

$Err:=MSG_MessageSize($msgfile;$HdrSize;$BdySize;$MsgSize)
$Err:=MSG_GetBody($msgfile;0;$BdySize;$BodyContent)
```

POP3_Delete

POP3_Delete (pop3_ID ; primerMsg ; ultimoMsg) -> resultado

Parámetro	Tipo		Descripción
pop3_ID	Entero largo	➡	Referencia de una conexión POP3
primerMsg	Entero largo	➡	Número del primer mensaje
ultimoMsg	Entero largo	➡	Número del último mensaje
resultado	Entero	⮌	Código de error

Descripción

En el intervalo de mensajes entre *primerMsg* y *ultimoMsg*, el comando *POP3_Delete* marca cada mensaje a borrar. La eliminación real de los mensajes no se produce hasta que se ejecute correctamente el comando *POP3_Logout*. Si la sesión actual se interrumpe por cualquier razón (tiempo de espera, fallo de la red, etc.) antes de llamar al comando *POP3_Logout*, los mensajes marcados para eliminación permanecerán en el servidor POP3.

pop3_ID es una referencia entero largo de una sesión abierta creada con *POP3_Login*.

primerMsg es el número del primer mensaje a eliminar.

ultimoMsg es el número del último mensaje a eliminar.

Nota: los comandos *POP3_Delete*, *POP3_MsgLstInfo* y *POP3_MsgLst* no devuelven un error si *primerMsg* es mayor que *ultimoMsg*. En caso de que esto ocurra, el comando no hace nada.

POP3_DownLoad

POP3_DownLoad (pop3_ID ; numMsg ; encabSolo ; nomArchivo) -> resultado

Parámetro	Tipo		Descripción
pop3_ID	Entero largo	➡	Referencia de una conexión POP3
numMsg	Entero largo	➡	Número del mensaje
encabSolo	Entero	➡	0 = Mensaje entero, 1 = Encabezado únicamente
nomArchivo	Texto	➡	Nombre del archivo local
		⬅	Nombre del archivo local utilizado
resultado	Entero	↩	Código de error

Descripción

El comando *POP3_DownLoad* está diseñado para recuperar un mensaje de un servidor POP3 mediante la descarga en un archivo en disco. Todo mensaje POP3 que contenga archivos adjuntos o cuyo tamaño es mayor que 32K debe ser descargado con este comando. Los archivos adjuntos sólo pueden extraerse de los mensajes recuperados de esta manera.

pop3_ID es una referencia entero largo a una sesión abierta creada con *POP3_Login*.

numMsg es un valor entero largo que indica cual mensaje en el buzón recuperar. *numMsg* representa la posición de un mensaje dentro de la lista actual de mensajes. El *numMsg* de un mensaje no es un valor estable, difiere de una sesión a otra en función de la eliminaciones.

encabSolo es un valor entero que indica si se debe recuperar todo el mensaje o sólo la información del encabezado.

nomArchivo contiene el nombre del archivo y la ruta de acceso opcional donde desea guardar el mensaje. Este valor se puede especificar de tres maneras diferentes:

""	= Guarda el archivo en la carpeta definida por *POP3_SetPrefs*, con el nombre "temp1" (si un archivo con el mismo nombre ya existe, se prueba con los nombres "temp2", "temp3", etc.)
"nomArchivo"	= Guarda el archivo en la carpeta definida por *POP3_SetPrefs*, con el nombre *nomArchivo*
"Ruta:nomArchivo"	= Guarda el archivo en la ruta especificada por *nomArchivo*

En los dos primeros casos, si *POP3_SetPrefs*, no especifica ninguna carpeta, el mensaje se guardará en la misma carpeta que la estructura de la base de datos (con 4D monopuesto) o en la carpeta 4D Client (con 4D Server). Después de que el archivo se guarde en el disco, el nombre final del archivo se devuelve en el parámetro *nomArchivo*. Si intenta llamar a *POP3_Download* con un *nomArchivo* que ya existe en la carpeta de descarga, el nombre se incrementa numéricamente y su nuevo valor como se guarda en el disco será devuelto a la variable *nomArchivo*.

POP3_GetMessage

POP3_GetMessage (pop3_ID ; numMsg ; offset ; longitud ; textoMsg) -> resultado

Parámetro	Tipo		Descripción
pop3_ID	Entero largo	→	Referencia de una conexión POP3
numMsg	Entero largo	→	Número del mensaje
offset	Entero largo	→	Carácter a partir del cual comenzar la recuperación
longitud	Entero largo	→	Número de caracteres a reenviar
textoMsg	Texto	←	Texto del mensaje
resultado	Entero	↩	Código de error

Descripción

El comando *POP3_GetMessage* devuelve el texto completo del mensaje identificado por *numMsg* en el buzón referenciado por *pop3_ID*. A menos que se especifique lo contrario por el comando *POP3_SetPrefs*, se eliminan los caracteres de salto de línea en el mensaje. El comando *POP3_GetMessage* devuelve todo el mensaje, incluyendo la información de encabezado.

pop3_ID es una referencia entero largo a una sesión abierta creada con *POP3_Login*.

numMsg es un valor entero largo que indica cual mensaje en el buzón recuperar. *numMsg* representa la posición de un mensaje dentro de la lista actual de mensajes. El *numMsg* de un mensaje no es un valor estable, difiere de una sesión a otra en función de las eliminaciones.

offset es un valor entero largo que indica la posición del carácter a partir del cual comenzar la lectura. En la mayoría de los casos pase cero en este parámetro.

El parámetro *longitud* es un valor entero largo que indica el número de caracteres a recuperar más allá de la posición *offset*. Dado que la longitud máxima de este parámetro está limitada a 32.000 caracteres por razones históricas, el parámetro *longitud* debe ser inferior a 32.000. Los mensajes cuyo tamaño sea mayor que 32 K debe recuperarse en el disco utilizando el comando *POP3_Download*.

textoMsg es una variable texto que recibe el texto recuperado.

POP3_GetPrefs

POP3_GetPrefs (retornoLinea ; carpetaMsg ; carpetaDocsAdj) -> resultado

Parámetro	Tipo		Descripción
retornoLinea	Entero	⬅	0 = No retirar los retornos de línea, 1 = Retirar los retornos de línea
carpetaMsg	Texto	⬅	Ruta de acceso a la carpeta de mensajes ("" = sin cambios)
carpetaDocsAdj	Texto	⬅	Ruta de acceso a la carpeta de los documentos adjuntos ("" sin modificación)
resultado	Entero	↩	Código de error

Descripción

El comando *POP3_GetPrefs* devuelve las preferencias actuales de los comandos POP3. Las preferencias se devuelven en las variables que figuran en los parámetros.

El parámetro *retornolinea* devuelve la configuración actual de la opción de eliminación de los retornos de línea.

El parámetro *carpetaMsg* es una variable texto que devuelve la ruta de acceso local a la carpeta en la que se guardan por defecto los mensajes recuperados.

El parámetro *carpetaDocAdj* es una variable texto que devuelve la ruta de acceso local a la carpeta en la que se guardan por defecto los documentos adjuntos extraídos de los mensajes.

Nota de compatibilidad (versión 6.8.1): el parámetro *carpetaDocAdj* del comando *POP3_GetPrefs* es opcional, por lo tanto, le recomendamos no pasar este parámetro, puesto que ya no se utiliza. Tenga en cuenta que este parámetro no afecta a los comandos POP3, ya que sólo es utilizado por los comandos MSG.

POP3_Login

POP3_Login (nomServidor ; nomUsuario ; contraseña ; aPOP ; pop3_ID {; paramSesion}) -> resultado

Parámetro	Tipo		Descripción
nomServidor	Cadena	➡	Nombre o dirección IP del servidor de correo POP3
nomUsuario	Cadena	➡	Nombre del usuario
contraseña	Cadena	➡	Contraseña
aPOP	Entero	➡	0 = Conexión texto plano, 1 = Conexión APOP
pop3_ID	Entero largo	⬅	Referencia a esta conexión POP3
paramSesion	Entero largo	➡	1 = Use SSL, 0 se omite = No utilizar SSL
resultado	Entero	↩	Código de error

Descripción

El comando *POP3_Login* conecta al usuario definido por *nomUsuario* y *contraseña* al servidor de correo POP3. Si *aPOP* vale 1, el mecanismo APOP (RFC#1321) se utiliza para la conexión. Si *aPOP* es cero o está vacío, la conexión se efectúa normalmente, con una contraseña en texto plano. pop3_Login devuelve un número de identificación para la conexión (*pop3_ID*) que los comandos posteriores deberán utilizar.

Atención: los servidores POP3 no fueron diseñados para ser accesibles de forma interactiva. Una vez conectado a un servidor, debe realizar las acciones necesarias y luego desconectarse del servidor tan pronto como sea posible. Entre las llamadas de *POP3_Login* y *POP3_Logout*, su procedimiento no debe participar en pantallas de usuario interactivo. Un servidor POP3 desconectará automáticamente todas las reuniones que no presenten actividad durante un período de tiempo determinado. De acuerdo con el RFC para POP3, el periodo de inactividad autorizado es de mínimo 30 minutos. Sin embargo, nuestra experiencia ha demostrado que la mayoría de servidores desconectan a los usuarios inactivos después de un período mucho más corto de tiempo.

Cada comando que interactúa con el servidor POP3 provoca un reinicio de su contador de inactividad. Si el servidor interrumpe la conexión antes de que haya llamado a *POP3_Logout*, todas las eliminaciones que haya realizado se cancelan.

nomServidor es el nombre o la dirección IP del servidor de correo POP3. Se recomienda utilizar el nombre del servidor, si es necesario, puede utilizar una dirección IP.

nomUsuario es el nombre del usuario del servidor de correo POP3. El parámetro *nomUsuario* no debe contener el dominio. Por ejemplo, para la dirección "jack@4d.com", el *nomUsuario* es "jack".

contraseña es la contraseña correspondiente a *nomUsuario* en el servidor de correo POP3.

El parámetro *aPOP* es un valor entero que indica si el mecanismo APOP debe ser utilizado para la conexión. Pase 1 para utilizar el mecanismo APOP. Pase cero para efectuar la conexión con contraseña en texto plano. El valor por defecto es cero.

pop3_ID es una variable de tipo entero largo en la que se devuelve una referencia a la conexión que se acaba de establecer. La variable se utiliza en todos los comandos posteriores que realizan acciones relacionadas con esta sesión.

El parámetro opcional *paramSesion* permite activar el protocolo SSL para la conexión:

- Si pasa 1, la conexión al servidor POP3 se efectúa en SSL (modo síncrono),
- Si pasa 0 u omite este parámetro, la conexión se efectúa en modo estándar, no seguro.

POP3_Logout

POP3_Logout (pop3_ID) -> resultado

Parámetro	Tipo		Descripción
pop3_ID	Entero largo		Referencia de una conexión POP3
			0 = Desconexión éxitosa
resultado	Entero		Código de error

Descripción

El comando *POP3_Logout* cierra la sesión POP3 referida por la variable *pop3_ID*. Si la desconexión se ha realizado correctamente, el valor 0 (cero) se devuelve en el parámetro *pop3_ID*.

Desconectarse del servidor POP3 indica al servidor que realice las eliminaciones solicitadas durante la sesión. Para deshacer una eliminación llame al comando *POP3_Reset* antes de *POP3_Logout*.

pop3_ID contiene la referencia de una sesión abierta con *POP3_Login*.

POP3_MsgInfo

POP3_MsgInfo (pop3_ID ; numMsg ; tamMsg ; IDUnico) -> resultado

Parámetro	Tipo		Descripción
pop3_ID	Entero largo	→	Referencia de una conexión POP3
numMsg	Entero largo	→	Número del mensaje
tamMsg	Entero largo	←	Tamaño del mensaje
IDUnico	Cadena	←	ID único de un mensaje en el servidor
resultado	Entero	↩	Código de error

Descripción

El comando *POP3_MsgInfo* devuelve el tamaño y el ID único del mensaje identificado por *numMsg* en el buzón referenciado por *pop3_ID*.

pop3_ID es una referencia entero largo a una sesión abierta creada con *POP3_Login*.

numMsg es un valor entero largo que indica cuál es el mensaje en el buzón del cual desea recuperar información. *numMsg* representa la posición de un mensaje dentro de la lista actual de mensajes. El *numMsg* de un mensaje no es un valor estable, difiere de una sesión a otra en función de las eliminaciones..

tamMsg devuelve el tamaño del mensaje.

IDunico es una variable entero largo que indica el identificador único del mensaje en el servidor. *IDunico* es un valor asignado al mensaje por el servidor POP3. Este valor no cambiará de una sesión a otra de la misma manera como *numMsg*. El valor *IDunico* es una buena referencia para comprobar si su base de datos ya ha descargado un mensaje desde el servidor.

POP3_MsgLst

POP3_MsgLst (pop3_ID ; inicio ; fin ; arrayEncabMsg ; arrayNumMsg ; arrayIDMsg ; arrayValores) -> resultado

Parámetro	Tipo		Descripción
pop3_ID	Entero largo	→	Referencia de una conexión POP3
inicio	Entero largo	→	Número del primer mensaje
fin	Entero largo	→	Número del último mensaje
arrayEncabMsg	Array cadena	→	Array de encabezados a recuperar
arrayNumMsg	Array entero largo	←	Array de los números de mensajes
arrayIDMsg	Array cadena	←	Array alfanumérico de los ID únicos
arrayValores	Array alfa 2D, Array texto 2D	←	Array 2D de los valores de los encabezados
resultado	Entero	↩	Código de error

Descripción

El comando *POP3_MsgLst* se utiliza para obtener información específica del contenido de los buzones. *arrayValores* es un array de dos dimensiones que recibe los datos de cada encabezado especificado en *arrayEncabMsg*. Cada encabezado solicitado tendrá un array correspondiente en la primera dimensión de *arrayValores*.

Este comando permite al usuario solicitar columnas específicas de la lista de mensajes. Este comando sólo puede devolver los valores de los encabezados, no se puede utilizar para recuperar el cuerpo de un mensaje.

Nota: los encabezados de correo pueden incluir caracteres extendidos, puede automatizar su gestión utilizando el comando *POP3_Charset*.

Ejemplo

```
aHeaders{1}:="Date:"
aHeaders{2}:="From:"
aHeaders{3}:="Subject:"
POP3_MsgLst(◇POP3_ID;vStart;vEnd;aHeaders;aMsgNum;aUIDs;aValues)
aValues{1}{1}por ejemplo"Jueves, 19 de noviembre 1998 00:24:02 -0800"
aValues{2}{1}por ejemplo"Jack@4d.com"
aValues{3}{1}por ejemplo"Llame a su esposa"
```

Los errores se manejan de la siguiente forma:

1) Sólo se devuelven los códigos de error relacionados con la comunicación. Si el comando no puede completar su tarea debido a un error (de la red, la sintaxis, el servidor, etc) luego se devuelve el código de error correspondiente.

2) Si un mensaje dentro del rango especificado de mensajes no existe o contiene un error:
-- No se crea ningún elemento de array para ese mensaje.
-- No se devuelve ningún código de error

3) La imposibilidad de localizar a uno o varios encabezados en un mensaje no constituye un error:
-- Un elemento de array es creado para el mensaje
-- Los elementos de array "número" e "ID" contienen los valores apropiados
-- Para cada encabezado que no se encuentre en el mensaje, se envía una cadena vacía al elemento de array
-- Ningún código de error se devolverá

Nota: los comandos *POP3_Delete*, *POP3_MsgLstInfo* y *POP3_MsgLst* no devuelven un error si el *primerMsg* es mayor que el *ultimoMsg*. En caso de que esto ocurra, el comando no hace nada.

POP3_MsgLstInfo

POP3_MsgLstInfo (pop3_ID ; primerMsg ; ultimoMsg ; arrayTam ; arrayNumMsg ; arrayIDMsg) -> resultado

Parámetro	Tipo		Descripción
pop3_ID	Entero largo	→	Referencia de una conexión POP3
primerMsg	Entero largo	→	Número del primer mensaje
ultimoMsg	Entero largo	→	Número del último mensaje
arrayTam	Array entero largo	←	Array de los tamaños
arrayNumMsg	Array entero largo	←	Array de los números de mensajes
arrayIDMsg	Array cadena	←	Array de los ID únicos
resultado	Entero	↩	Código de error

Descripción

El comando *POP3_MsgLstInfo* devuelve información sobre un conjunto de mensajes en un buzón. La información se devuelve en tres arrays, cada elemento de los arrays corresponde a un mensaje. Se devuelve información sobre el tamaño de cada mensaje, el número de mensaje y el ID único del mensaje. Los arrays pasados como parámetros deben ser declarados, aunque pueden ser de cualquier tamaño. El comando *POP3_MsgLstInfo* redimensiona cada array al número de mensajes recuperados.

El comando *POP3_MsgLstInfo* no devolverá un número de error si no puede recuperar la información sobre un mensaje dentro de la lista de mensajes actual. Si se detecta un error, no se crea ningún elemento en los arrays para el mensaje del problema. Si el comando lee todos los mensajes con éxito, *arrayNumMsg* debe contener valores numéricos en un orden secuencial. En caso de problemas, se perderá la secuencia numérica de *arrayNumMsg*.

pop3_ID es una referencia entero largo a una sesión abierta creada con *POP3_Login*.

primerMsg es un número entero largo que especifica el número del primer mensaje a examinar. El número de mensaje es un valor que representa la posición de un mensaje en la lista de todos los mensajes en el buzón identificado por *pop3_ID*.

ultimoMsg es un número entero largo que indica el número del último mensaje a examinar. El número de mensaje es un valor que representa la posición de un mensaje en la lista de todos los mensajes en el buzón identificado por *pop3_ID*.

arrayTam recibe el tamaño de cada mensaje entre *primerMsg* y *ultimoMsg*.

arrayNumMsg devuelve los números de los mensajes entre *primerMsg* y *ultimoMsg*.

arrayIDMsg es un array cadena o texto devuelto contiene el identificador único de los mensajes entre *primerMsg* y *ultimoMsg* .

Nota: los comandos *POP3_Delete*, *POP3_MsgLstInfo* y *POP3_MsgLst* no devuelven un error si el *primerMsg* es mayor que el *ultimoMsg*. En caso de que esto ocurra, el comando no hace nada.

POP3_Reset

POP3_Reset (pop3_ID) -> resultado

Parámetro	Tipo		Descripción
pop3_ID	Entero largo	→	Referencia de una conexión POP3
resultado	Entero	↩	Código de error

Descripción

El comando *POP3_Reset* permite anular la opción de eliminar, aplicada a los mensajes durante la sesión actual.

Nota: el comando **POP3_Delete** no borra inmediatamente los mensajes, su acción consiste únicamente en marcar los mensajes a borrar. Los mensajes en un servidor POP3 sólo se borran en el momento de una desconexión exitosa (*POP3_Logout*).

pop3_ID contiene una referencia a una sesión abierta con *POP3_Login*.

POP3_SetPrefs

POP3_SetPrefs (retornoLinea ; carpetaMsg ; carpetaDocsAdj) -> resultado

Parámetro	Tipo		Descripción
retornoLinea	Entero	→	0 = No eliminar los retornos de línea, 1 = Retirar los retornos de línea, -1 = No cambiar
carpetaMsg	Texto	→	Ruta de acceso a la carpeta de mensajes ("" = no cambiar)
carpetaDocsAdj	Texto	→	Ruta de acceso a la carpeta de documentos adjuntos ("" = no cambiar)
resultado	Entero	←	Código de error

Descripción

El comando *POP3_SetPrefs* define las preferencias para todos los comandos POP3.

retornoLinea es un valor entero que especifica como tratar los caracteres de retorno de línea en los mensajes guardados. La mayoría de los servidores POP3 combinan los caracteres retorno de carro y salto de línea para indicar el final de una línea. Las aplicaciones de Macintosh sólo requieren de un retorno de carro. Esta opción permite a los usuarios retirar los caracteres retorno de línea del texto de los mensajes. Un valor de cero deja los mensajes recuperados en el formato del servidor POP3. Un valor de uno retira los caracteres de salto de línea de los mensajes recuperados. Un valor de -1 dejará esta preferencia, como se estableció previamente. Por defecto, esta opción está en 1 y los retornos de línea se eliminan automáticamente en los mensajes.

carpetaMsg indica la ruta de acceso local de la carpeta en la que los mensajes recuperados con el comando *POP3_Download* se almacenan por defecto.

Nota de compatibilidad (versión 6.8.1): los parámetros *retornoLinea* y *carpetaMsg* se aplicaron previamente a MSG_Commands. Este ya no es el caso cuando se utiliza el comando *MSG_SetPrefs*.

carpetaDocsAdj es un valor de texto que contiene la ruta de acceso local de la carpeta en la que se guardan los archivos adjuntos cuando el comando **MSG_Extract** extrae los documentos adjuntos del cuerpo del mensaje.

Nota de compatibilidad (versión 6.8.1): el parámetro *carpetaDocsAdj* también se encuentra en *POP3_SetPrefs* y *MSG_SetPrefs* por lo tanto usted puede modificarlo utilizando cualquiera de estos dos comandos. Se recomienda el uso del comando *MSG_SetPrefs*; el parámetro *POP3_SetPrefs*, utilizado por razones de compatibilidad, no se utilizará en el futuro. El parámetro del comando *POP3_SetPrefs* es opcional; por lo tanto, le recomendamos no utilizar este parámetro. Esta recomendación se aplica también al comando *POP3_GetPrefs*.

POP3_UIDToNum

POP3_UIDToNum (pop3_ID ; IDUnico ; numMsg) -> resultado

Parámetro	Tipo		Descripción
pop3_ID	Entero largo	⇒	Referencia de una conexión POP3
IDUnico	Cadena	⇒	ID único de un mensaje en el servidor
numMsg	Entero largo	⇐	Número del mensaje
resultado	Entero	↩	Código de error

Descripción

El comando *POP3_UIDToNum* devuelve el número de mensaje *numMsg* correspondiente al mensaje designado por *IDUnico* en el buzón referenciado por *pop3_ID*. El *numMsg* de un mensaje es un valor fluctuante relativo a los otros mensajes de la lista, este comando devuelve la posición actual de un mensaje cuya información puede haber sido recuperada durante una sesión POP3 anterior.

pop3_ID es una referencia entero largo a una sesión abierta creada con *POP3_Login*.

IDUnico es un valor cadena que contiene el identificador único de un mensaje para localizarlo en el servidor POP3. Este comando devuelve el número de mensaje *numMsg* correspondiente al mensaje designado por *IDUnico* en el buzón referenciado por *pop3_ID*. Una vez encontrado, la posición actual del mensaje en la lista será devuelta en *numMsg*.

numMsg es un entero largo devuelto que contiene el número de mensaje actual (su posición en la lista de mensajes actuales) del elemento identificado por *IDunico*. Si no se puede encontrar en el servidor el *IDunico*, numMsg devuelve cero y no se devuelve ningún error.

POP3_VerifyID

POP3_VerifyID (pop3_ID) -> resultado

Parámetro	Tipo		Descripción
pop3_ID	Entero largo	→	Referencia de una conexión POP3
		←	0 = Desconexión éxitosa
resultado	Entero	↩	Código de error

Descripción

Un servidor POP3 desconecta automáticamente las sesiones que no muestran actividad en un período de tiempo determinado por el administrador. Cada comando que interactúa con el servidor POP3 reinicia en cero el contador de inactividad. El comando *POP3_VerifyID* provoca igualmente la reinicialización en cero del contador de inactividad de la sesión POP3 especificada. Esto permite al usuario mantener una sesión activa si existe la posibilidad de que la sesión sobrepase el timeout.

Cuando se ejecuta, el comando *POP3_VerifyID* verifica que la conexión no se haya cerrado. Si la sesión aún está abierta, el comando le indica al servidor POP3 reiniciar en cero el contador de inactividad para la sesión. Si la conexión ya ha cerrado, *POP3_VerifyID* devuelve el error correspondiente y libera la memoria usada por la sesión POP3 y devuelve cero en el parámetro *pop3_ID*.

pop3_ID es una referencia entero largo de una sesión abierta creada con *POP3_Login*.

IC Send Mail

- Envío de correo, Presentación
- SMTP_AddHeader
- SMTP_Attachment
- SMTP_Auth
- SMTP_Bcc
- SMTP_Body
- SMTP_Cc
- SMTP_Charset
- SMTP_Clear
- SMTP_Comments
- SMTP_Date
- SMTP_Encrypted
- SMTP_From
- SMTP_GetPrefs
- SMTP_Host
- SMTP_InReplyTo
- SMTP_Keywords
- SMTP_MessageID
- SMTP_New
- SMTP_QuickSend
- SMTP_References
- SMTP_ReplyTo
- SMTP_Send
- SMTP_Sender
- SMTP_SetPrefs
- SMTP_Subject
- SMTP_To

Envío de correo, Presentación

El protocolo SMTP (Simple Mail Transfer Protocol) es el principal protocolo de transferencia de correo usado en Internet. Con 4D Internet Commands, los desarrolladores pueden crear mensajes electrónicos simples con un solo comando, o mensajes complejos con una serie de comandos. Los comandos SMTP permiten a los desarrolladores controlar todas las partes de un mensaje electrónico, incluyendo los encabezados de Respuesta y Remitente, documentos adjuntos, comentarios y referencias.

Los comandos 4D y 4D Internet Commands permiten a los desarrolladores crear bases de datos muy poderosas con la posibilidad de enviar mensajes y archivos adjuntos en Internet. Algunos ejemplos de cómo el conjunto de comandos SMTP podría mejorar sus bases de datos son:

- Automatización del envío de informes o documentos creados en 4D.
- Creación de aplicaciones 4D que informen a los desarrolladores eventos específicos (por ejemplo ON ERR CALL("Mail_Error"))
- Envío de correos automáticamente a las personas en todo el país

Hay un número ilimitado de usos para el conjunto de comandos SMTP. Estos comandos, junto con los comandos POP3 (recuperación de archivos y documentos adjuntos), FTP y TCP ofrecen al desarrollador las herramientas para aumentar considerablemente la capacidad de las comunicaciones de sus bases de datos 4D.

Dos métodos de creación de un mensaje electrónico

Los comandos SMTP permiten enviar correo electrónico de dos formas diferentes, llamados "simple" y "complejo". El método "simple" implica un solo comando, *SMTP_QuickSend*, que acepta todos los parámetros necesarios para el direccionamiento y envío de un mensaje.

La mayoría de los correos electrónicos enviados en todo el mundo es bastante simple en su construcción; alguien "aquí" quiere enviar un "mensaje" de algún tipo a alguien "allá" con respecto a algún "tema". Si fuera una carta de papel, usted escribiría todo lo anterior, pondría una estampilla y la dirección en el sobre y luego lo llevaría a la oficina de correos para la entrega. Con SMTP_QuickSend, puede especificar el campo "De", "Para", "Asunto" y "Cuerpo del mensaje" en un solo comando.

Sin embargo, algunos mensajes pueden necesitar parámetros más complejos. Por ejemplo, supongamos que la carta mencionada anteriormente debe enviarse con copias a otras partes interesadas, o tal vez deba añadir un archivo adjunto, como su informe anual. En esos casos, debe fotocopiar la carta y los informes impresos y hacer un sobre para cada destinatario. Los comandos SMTP de 4D simplifican la distribución, dándole el control de todos los aspectos de la transmisión de correo electrónico. Múltiples archivos adjuntos, envió de copias y de copias ocultas, así como cualquier especificación de encabezado de correo pueden ser manejados a través de las capacidades integradas de los comandos SMTP.

Entrega de correo

Uno de los conceptos fundamentales en la comprensión del funcionamiento de los comandos SMTP se refiere al modo de distribución del correo a sus destinatarios. Los comandos SMTP no entregan directamente el correo a cada destinatario. Los comandos hacen la composición y dan el formato apropiado a su correo y entregan los resultados al servidor SMTP especificado por el comando SMTP_host. El servidor SMTP es generalmente una máquina de su empresa o de su proveedor de servicios de Internet. El servidor SMTP decide la ruta de entrega óptima para su correo electrónico y programa su distribución en función de los parámetros configurados por el administrador de correo.

Requisitos mínimos para enviar un mensaje SMTP complejo

Para una buena trasmisión de los mensajes electrónicos usando los comandos SMTP, los comandos deben estar correctamente configurados. Los siguientes comandos representan el mínimo para que la entrega de correo electrónico sea exitosa:

- *SMTP_New*
  Crea el espacio en la memoria para el nuevo mensaje y devuelve una referencia que se utilizará en los comandos posteriores.

- *SMTP_Host*
  Especifica el servidor SMTP donde entrega el mensaje

- *SMTP_From*
  Por lo menos una dirección en este encabezado

- *SMTP_To*
  Por lo menos una dirección en este encabezado

- *SMTP_Send*
  Envía el mensaje

- *SMTP_Clear*
  Borra toda la memoria utilizada durante la creación del mensaje

Si sólo se ejecutan estos comandos, un mensaje sin asunto ni cuerpo se hubiera enviado. Esto no es particularmente útil e ilustra la necesidad de especificar detalles adicionales con el fin de comunicar de manera eficaz su mensaje.

Unicode por defecto (4D v14)

A partir de 4D v14, los campos "objeto" (*subject*) y "cuerpo" (*body*) de los comandos SMTP utilizan por defecto el conjunto de caracteres UTF-8. Este conjunto de caracteres será interpretado correctamente por casi todos los clientes de correo electrónico. Este funcionamiento simplifica en gran medida el uso de los comandos SMTP y ahora limita la utilidad de los comandos **SMTP_Charset** y **SMTP_SetPrefs**.

SMTP_AddHeader

SMTP_AddHeader (smtp_ID ; nomEncabezado ; textoEncab {; eliminarOpcion}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
nomEncabezado	Cadena	➡	Nombre del encabezado
textoEncab	Texto	➡	Texto del encabezado
eliminarOpcion	Entero	➡	0 = Añadir 1 = Reemplazar todos los encabezados con 'nombreEncabezado', 2 = Eliminar todos los encabezados llamados 'nomEncab'
resultado	Entero	↩	Código de error

Descripción

El comando **SMTP_AddHeader** permite a los usuarios añadir sus propios encabezados al mensaje referenciado por smtp_ID. Además de los encabezados estándar generados por los comandos Internet de 4D, hay dos categorías de encabezados adicionales: los encabezados "usuario" (definidos por el usuario) y los encabezados "extendidos". El comando **SMTP_AddHeader** permite al usuario añadir el nuevo encabezado y los datos a asociar a él.

Encabezados "extendidos": estos encabezados han sido oficialmente reconocidos por el NIC y se definieron después de las especificaciones SMTP iniciales. Estos encabezados a menudo tienen una función específica afectando el comportamiento de las diferentes aplicaciones de software. Los encabezados "extendidos" nunca comienzan por la letra "X".

Encabezados "usuario": el protocolo SMTP permite que cualquiera pueda crear sus propias definiciones de encabezado. Todos los encabezados definidos por el usuario deben comenzar por los caracteres "X-" para evitar todo conflicto con un futuro encabezado "extendido". Los encabezados "usuario" son particularmente útiles cuando usted controla ambos extremos de la comunicación.

Los encabezados "usuario" permiten al desarrollador almacenar datos que se pueden extraer fácilmente con el comando POP3 *MSG_FindHeader*. Por ejemplo, puede crear un encabezado llamado "X-001001", que contiene el valor en el campo 01 del archivo 01. Puede añadir un número ilimitado de encabezados a un mensaje. Los encabezados "usuario" le dan al usuario la posibilidad de añadir la información más fácil de extraer sin necesidad de analizar el cuerpo del mensaje.

smtp_ID es la referencia entero largo de un mensaje electrónico creado con el comando *SMTP_New*.

nomEncab contiene el nombre del encabezado a añadir.

textoEncab contiene la información que se asignará en el área de encabezado identificada por *nomEncab*.

Atención: el texto no debe contener retornos de línea (ASCII = 10). Un retorno de línea significa el final de la sección de encabezado y el comienzo del cuerpo. Los encabezados posteriores podrían ser considerados como el cuerpo del texto y no ser reconocidos correctamente por el software del servidor o del cliente. Para obtener más información sobre los encabezados, consulte la RFC#822.

Nota: el comando no hace nada si *nomEncab* o *textoEncab* es una cadena vacía (sin encabezado añadido).

eliminarOpcion permite precisar si desea eliminar el encabezado actual. Si pasa cero, el encabezado *nomEncab* se añade al mensaje. Si pasa 1, todos los encabezados del mensaje se remplazan por el encabezado *nomEncab*. En este caso, si *nomEncab* es una cadena vacía, todos los encabezados se eliminarán. Si pasa 2, todos los encabezados *nomEncab* se eliminan del mensaje.

Nota: a partir de la versión 14 de 4D Internet Commands, cuando desee enviar un mensaje en formato HTML, ya no es necesario cambiar el encabezado "content-type" utilizando **SMTP_AddHeader**. Puede definir el formato HTML directamente utilizando el comando **SMTP_Body**, en cuyo caso el "Content-Type" se definirá automáticamente como "text/html;charset=utf-8" (de lo contrario, el "Content-Type" está configurado por defecto como "text/plain;charset=utf-8"). Sin embargo, para necesidades específicas, puede siempre "forzar" el campo "Content-Type" con **SMTP_AddHeader**. En este caso, asegúrese de especificar el conjunto de caracteres (normalmente "charset=utf-8", ya que, por defecto, 4D IC siempre envía el cuerpo como UTF-8).

SMTP_Attachment

SMTP_Attachment (smtp_ID ; nomArchivo ; tipoCod ; eliminarOpcion {; idAdjunto {; contentType}}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
nomArchivo	Texto	➡	Nombre del archivo a adjuntar
tipoCod	Entero	➡	0 = No codificar (sólo envía el DataFork) ±1 = BinHex ±2 = Base64; (sólo envía el DataFork) ±3 = AppleSingle ±4 = AppleDouble ±5 = AppleSingle Y Base64 ±6 = AppleDouble Y Base64 ±7 = UUEncode
eliminarOpcion	Entero	➡	0 = Añadir a la lista existente, 1 = Reemplazar todos los adjuntos por nomArchivo, 2 = Eliminar sólo este adjunto
idAdjunto	Texto	➡	ID del archivo adjunto(mensajes HTML únicamente)
contentType	Texto	➡	Valor del tipo de contenido a definir
resultado	Entero	↩	Código de error

Descripción

El comando **SMTP_Attachment** permite añadir archivos binarios o de texto a su mensaje en formato MIME. Este comando puede llamarse varias veces para adjuntar varios documentos a un mensaje. Si pasa un valor mayor que cero en el parámetro *tipoCod*, este comando realizará la codificación en el momento del envío del mensaje.

smtp_ID es la referencia entero largo de un mensaje electrónico creado con el comando *SMTP_New*.

nomArchivo designa el nombre del archivo a adjuntar al mensaje. Este valor se puede especificar de dos maneras:

- nomArchivo = busca el nombre del archivo en el mismo directorio de la estructura de la base de datos
- Ruta:nombreArchivo = ruta de acceso completa incluyendo el nombre del archivo

Nota: en 4D Internet Commands v16 R2 y superiores, los nombres de los archivos están limitados a 1.024 caracteres, cualquiera que sea el sistema operativo. Sin embargo, en las versiones de macOS 32 bits, 4D Internet Commands utiliza las APIs de gestión administración de archivos declarados obsoletos que ya no son mantenidas por Apple. En este entorno, dependiendo de la versión de macOS, los nombres de archivo pueden tener limitaciones más restrictivas en cuanto a longitud y caracteres admitidos. Recomendamos encarecidamente actualizar a 4D Internet Commands 64 bits tan pronto como sea posible.

tipoCod es un valor entero que indica qué tipo de codificación aplicar al archivo antes de integrarlo al mensaje. Si adjunta un archivo binario, debe utilizar un método de codificación adecuado (BinHex, AppleSingle). El método de codificación más común es BinHex.

Si pasa valores positivos en *tipoCod* el comando codifica automáticamente el archivo utilizando el método especificado cuando se envía el mensaje. La codificación de un archivo se produce en el momento en que se ejecuta el comando *SMTP_Send*. Si el archivo es grande la ejecución del comando *SMTP_Send* puede tardar unos momentos. Puede ahorrar tiempo cuando el mismo archivo se envía varias veces. En estos casos lo mejor es codificar el archivo una vez con el comando *IT_Encode* y luego adjuntar el archivo resultante a su mensaje utilizando el valor negativo de *tipoCod*. Un valor negativo en *tipoCod* no realizará ninguna codificación adicional, sino que el tipo de codificación se describe en el encabezado del archivo adjunto al mensaje. Esto informa al software de mensajería del destinatario la forma correcta de interpretar los datos adjuntos.

Nota: no puede pasar un elemento de array en el parámetro *tipoCod*.

eliminarOpcion es un parámetro entero opcional que especifica la forma de tratar los datos adjuntos.

- Un valor de cero añade el archivo adjunto a la lista actual de los archivos adjuntos.
- Un valor de 1 reemplaza todos los archivos adjuntos con el archivo *nomArchivo*. Si *nomArchivo* es una cadena vacía, todos los archivos adjuntos se eliminarán.
- Un valor de 2 eliminará sólo el archivo adjunto listado en *nomArchivo* de la lista de archivos adjuntos.

El parámetro *idAdjunto* asocia el archivo adjunto con una referencia definida en el cuerpo del mensaje utilizando la etiqueta HTML t**<img src=\"cid:ID\">**. Esto significa que el contenido de los archivos, por ejemplo una imagen, se pueden mostrar en el mensaje en el cliente de correo electrónico.
Este funcionamiento sólo es compatible con los mensajes en HTML. También tenga en cuenta que el resultado final puede variar dependiendo del cliente de correo electrónico.

El parámetro opcional *contentType* define explícitamente el tipo de contenido del archivo adjunto. De forma predeterminada, si este parámetro se omite o contiene una cadena vacía, 4DIC define automáticamente el tipo de contenido del archivo adjunto en función de su extensión. Se aplican las siguientes reglas:

Extensión	Tipo de contenido
jpg, jpeg	image/jpeg
png	image/png
gif	image/gif
pdf	application/pdf
doc	application/msword
xls	application/vnd.ms-excel
ppt	application/vnd.ms-powerpoint
zip	application/zip
gz	application/gzip
json	application/json
js	application/javascript
ps	application/postscript
xml	application/xml
htm, html	text/html
mp3	audio/mpeg
other	application/octet-stream

En *contentType*, puede pasar una cadena que defina el tipo de contenido para el archivo (tipo MIME), por ejemplo "video/mpeg". Este valor de tipo de contenido se establecerá para los datos adjuntos, independientemente de su extensión.

Nota: pase una cadena vacía ("") en el parámetro *idAdjunto* si no lo desea utilizar.

Ejemplo 1

Envío de un mensaje HTML con una imagen incluida:

```
$error:=SMTP_New($smtp_id)
$error:=SMTP_Host($smtp_id;"smtp.gmail.com")
$error:=SMTP_From($smtp_id;"henry@gmail.com")
$error:=SMTP_ReplyTo($smtp_id;"replies@gmail.com")
$error:=SMTP_Subject($smtp_id;"HTML Test  &  picture included")
$error:=SMTP_To($smtp_id;"john@4d.com";1)
$error:=SMTP_Body($smtp_id;"<html><B><I>Hello world in bold!</I></B> <img src=\"cid:myID123\">
(normal text)</HTML>";4)
$error:=SMTP_Attachment($smtp_id;"c:\\temp\\tulips.jpg";2;0;"myID123")
$error:=SMTP_Auth($smtp_id;"henry@gmail.com";"*******")
$error:=SMTP_Send($smtp_id;1)
$error:=SMTP_Clear($smtp_id)
```

Ejemplo 2

Queremos declarar sus archivos de configuración como archivos XML:

```
$path:=Get 4D folder(Database folder)+"Settings.mySettings"
$err:=SMTP_Attachment($smtp_id;$path;2;0;"myID123";"application/xml")
```

SMTP_Auth

SMTP_Auth (smtp_ID ; nomUsuario ; contraseña {; autMod}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
nomUsuario	Cadena	➡	Nombre del usuario para la autenticación SMTP
contraseña	Cadena	➡	Contraseña para la autenticación SMTP
autMod	Entero	➡	Modo de autenticación a utilizar: 0 o se omite = Modo definido por el servidor 1= PLAIN, 2 = LOGIN, 3 = CRAM-MD5
resultado	Entero	⮌	Código de error

Descripción

El comando *SMTP_Auth* permite el envío del mensaje referenciado por *smtp_ID* cuando un mecanismo de autenticación es requerido por el servidor SMTP. Este tipo de autenticación es requerida por algunos servidores SMTP con el fin de reducir el riesgo de que los mensajes se falsifiquen o que la identidad del remitente sea usurpada, particularmente con el fin de enviar spam.

Este comando se puede utilizar si la autenticación es necesaria o no, ya que sólo se ejecuta si *nomUsuario* y *contraseña* no son cadenas vacías.

smtp_ID es la referencia entero largo de un mensaje electrónico creado con el comando *SMTP_New*.

nomUsuario es el nombre del usuario en el servidor SMTP. El parámetro *nomUsuario* no debe contener el dominio. Por ejemplo, para la dirección "jack@4d.com", el *nomUsuario* es "Jack".

contraseña es la contraseña para *nomUsuario* en el servidor SMTP.

Nota: si *nomUsuario* y/o *contraseña* son cadenas vacías, el comando *SMTP_Auth* no se ejecuta.

El parámetro opcional *autMod* permite "forzar" el modo de autenticación utilizado. Puede pasar 0, 1, 2 ó 3 en este parámetro:

- Si pasa 0 (cero), el modo de autenticación utilizado por el comando *SMTP_Auth* será el modo más seguro soportado por el servidor (CRAM-MD5, LOGIN luego PLAIN),
- Si pasa 1, el método de autenticación utilizado será PLAIN,
- Si pasa 2, el método de autenticación utilizado será LOGIN,
- Si pasa 3, el método de autenticación utilizado será CRAM-MD5.
  Si omite *autMod*, por defecto se utiliza el valor 0. Si el método de autenticación solicitado por este parámetro no es compatible con el servidor SMTP, se devuelve un error.

Ejemplo

Este ejemplo permite enviar un mensaje con o sin autenticación, en función del contenido de campos especificados almacenados en la base 4D:

C_INTEGER($vError)

```
C_LONGINT($vSmtp_id)
C_TEXT($vAuthUserName;$vAuthPassword)

$vError:=SMTP_New($vSmtp_id)
$vError:=SMTP_Host($vSmtp_id;"wkrp.com")
$vError:=SMTP_From($vSmtp_id;"herb_tarlick@wkrp.com")
$vError:=SMTP_Subject($vSmtp_id;"Are you there?")
$vError:=SMTP_To($vSmtp_id;"Dupont@wkrp.com")
$vError:=SMTP_Body($vSmtp_id;"Podemos reunirnos?")

` Los campos se introducen si el servidor utiliza un mecanismo
` de autenticación. De lo contrario, se devuelven las cadenas vacías.
$vAuthUserName:=[Account]AuthUser
```

```
$vAuthPassword:=[Account]AuthPass

$vError:=SMTP_Auth($vSmtp_id;$vAuthUserName;$vAuthPassword)
$vError:=SMTP_Send($vSmtp_id)
$vError:=SMTP_Clear($vSmtp_id)
```

SMTP_Bcc

SMTP_Bcc (smtp_ID ; copiaDiscreta {; eliminarOpcion}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
copiaDiscreta	Texto	➡	Lista de direcciones
eliminarOpcion	Entero	➡	0 = Añadir, 1 = Reemplazar, 2 = Borrar
resultado	Entero	⮌	Código de error

Descripción

El comando *SMTP_Bcc* añade destinatarios en copia oculta al mensaje especificado por *smtp_ID*. No es obligatorio tener direcciones en el campo Cco:.

La única manera de mantener la confidencialidad de la lista de direcciones al enviar un correo a un grupo de personas es insertar las direcciones dentro del área de encabezados "Cco". Las direcciones listadas en este encabezado no se envían como parte del encabezado o cuerpo del mensaje. Las direcciones no podrán ser vistas por ningún destinatario del mensaje.

Un destinatario "Cco" puede ver todos los destinatarios en "Para" y "Cc", pero no los destinatarios "Cco". Generalmente, durante el envío de un correo masivo, todos los destinatarios deben estar en el encabezado "Cco". Esto evita que los mensajes recibidos tengan grandes listas de direcciones saturando el mensaje y que los usuarios accedan a las direcciones de los demás.

Otra razón para el uso de "Cco" es que muchas aplicaciones de correo tienen una función "Responder a todos", que añade todos los destinatarios en las secciones "Para" y "Cc" al mensaje de respuesta. Colocar todas las direcciones de los destinatarios en el encabezado "Cco" impedirá que los usuarios respondan a todas las personas que recibieron el mensaje original..

smtp_ID es la referencia entero largo de un mensaje creado con el comando *SMTP_New*.

copiaDiscreta es un valor texto que contiene una o más direcciones de correo.

eliminarOpcion es un valor entero que especifica si desea añadir o eliminar el encabezado "Bcc":

- Un valor de cero, añade el nuevo valor al campo "Bcc".
- Si pasa un valor de 1, el contenido del parámetro pasado reemplaza el contenido del encabezado existente. En este caso, si pasa una cadena vacía en *copiaDiscreta*, el encabezado "Bcc" se eliminará.
- Si pasa 2, el encabezado "Bcc" se borra del mensaje.
  eliminarOpcion es un parámetro opcional que por defecto tiene el valor cero.

Ejemplo

En este ejemplo se crea un mensaje y los elementos estáticos se definen fuera del alcance del bucle 'for'. Luego, para cada registro en la tabla [Personas], se añade una dirección a la lista de copias ocultas.

```
$error:=SMTP_From($smtp_id;"sales@massmarket.com")
$error:=SMTP_Subject($smtp_id;"¡Ofertas increíbles! ¡Sólo esta semana!")
$error:=SMTP_Body($smtp_id;$GenericBody)
For($i;1;Records in selection([Personas]))
   $error:=SMTP_Bcc($smtp_id;[Personas]Email;0)  `Añada esta dirección de correo electrónico a la lista BCC
   NEXT RECORD([Personas])
End for
$error:=SMTP_Send($smtp_id)  `Enviar el mensaje a todos
$error:=SMTP_Clear($smtp_id)
```

SMTP_Body

SMTP_Body (smtp_ID ; msgCuerpo {; opcion}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
msgCuerpo	Texto	➡	Cuerpo del mensaje
opcion	Entero	➡	0 = Remplazar (si msgCuerpo no está vacío), 1 = Eliminar, 2 = Añadir, 4 = Formato HTML (texto plano por defecto)
resultado	Entero	⬅	Código de error

Descripción

El comando **SMTP_Body** inserta el texto de *msgCuerpo* en el cuerpo principal del mensaje identificado por *smtp_ID*.
El *msgCuerpo* es el bloque principal de texto.

smtp_ID es la referencia entero largo de un mensaje electrónico creado con el comando **SMTP_New**.

msgCuerpo es un valor texto que contiene el cuerpo del mensaje.

Por defecto, el cuerpo del mensaje está codificado con UTF-8, lo que garantiza que los caracteres enviados serán interpretados correctamente por casi todos los clientes de correo electrónico. Si desea utilizar un conjunto de caracteres específico, consulte los comandos **SMTP_SetPrefs** y **SMTP_Charset**.

opcion se utiliza para gestionar la concatenación de varios cuerpos y para modificar el formato de los mensajes (texto o HTML):

- Si pasa cero, el contenido del parámetro *msgCuerpo* remplaza el cuerpo del mensaje presente, excepto si pasa una cadena vacía en *msgCuerpo*, en cuyo caso se utilizará el texto anterior.
- Si pasa 1, el contenido del parámetro *msgCuerpo* remplaza el cuerpo del mensaje presente. En este caso, si pasa una cadena vacía en *msgCuerpo*, se borra el cuerpo del mensaje existente.
- Si pasa 2, el contenido del parámetro *msgCuerpo* se anexará al cuerpo del mensaje, después de cualquier texto ya definido (concatenación)
  Si *opcion* se omite, por defecto se utiliza el valor 0.
- Si pasa 4, indica que el contenido de *msgCuerpo* es HTML (por defecto, *msgCuerpo* se envía como texto sin formato).

Para combinar el envío del mensaje en HTML con una opción de reemplazo, sólo puede sumar los valores. Por ejemplo, puede pasar 1+4 en *opcion* con el fin de sustituir el cuerpo y enviar el mensaje en HTML.

Ejemplo

Este es un ejemplo SMTP completo:

```
C_LONGINT($SMTP_ID)
C_BOOLEAN($SentOK;$OK)
$SentOK:=False  `Una bandera indica si es aplicable a todos los comandos
Case of
   :(Not(ERRCHECK("SMTP_New";SMTP_New($SMTP_ID))))
   :(Not(ERRCHECK("SMTP_Host";SMTP_Host($SMTP_ID;◇pref_Server))))
   :(Not(ERRCHECK("SMTP_From";SMTP_From($SMTP_ID;vFrom))))
   :(Not(ERRCHECK("SMTP_To";SMTP_To($SMTP_ID;vTo))))
   :(Not(ERRCHECK("SMTP_Cc";SMTP_Cc($SMTP_ID;vCC))))
   :(Not(ERRCHECK("SMTP_Bcc";SMTP_Bcc($SMTP_ID;vBcc))))
   :(Not(ERRCHECK("SMTP_Subject";SMTP_Subject($SMTP_ID;vSubject))))
   :(Not(ERRCHECK("SMTP_Comments";SMTP_Comments($SMTP_ID;"Sent via 4D"))))
   :(Not(ERRCHECK("SMTP_AddHeader";SMTP_AddHeader($SMTP_ID;"X-4Ddemo:";◇VERSION))))
```

```
    :(Not(ERRCHECK("SMTP_Body";SMTP_Body($SMTP_ID;vMessage))))
    :(Not(ERRCHECK("SMTP_Send";SMTP_Send($SMTP_ID))))
    Else
        $SentOK:=True  `mensaje compuesto y enviado con éxito
End case

If($SMTP_ID#0)  `Si un mensaje fue creado en memoria, debemos borrarlo ahora
    $OK:=ERRCHECK("SMTP_Clear";SMTP_Clear($SMTP_ID))
End if
```

Nota: para mayor información sobre el uso particular de la estructura **Case of**, consulte el **Anexo A: Consejos de programación**

A continuación está el código del método ***ERRCHECK***. Este método recibe dos parámetros, el nombre del comando ($Command), y el valor del error (pasado por la ejecución del comando en el parámetro del método). ***ERRCHECK*** devuelve un error booleano indicando si el comando devolvió el error cero. Si el error no es cero, el valor devuelto ($0) es false, de lo contrario es true.

```
C_TEXT(vErrorMsg)
$Command:=$1
$Error:=$2
$Result:=True
If($Error#0)
    $Result:=False
    If(◇SHOWERRORS)  `Booleano para determinar si mostrar los mensajes de error
        vErrorMsg:=IT_ErrorText($Error)
        ALERT("ERROR ---"+Char(13)+"Command: "+$Command+Char(13)+"Error
        Code"+String($Error)+Char(13)+"Description"+vErrorMsg)
    End if
End if
$0:=$Result
```

SMTP_Cc

SMTP_Cc (smtp_ID ; copiaCarbon {; eliminarOpcion}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
copiaCarbon	Texto	➡	Dirección electrónica o Lista de direcciones
eliminarOpcion	Entero	➡	0 = Añadir, 1 = Reemplazar, 2 = Borrar
resultado	Entero	⮌	Código de error

Descripción

El comando *SMTP_Cc* añade destinatarios en "cc" (carbon copy) al mensaje especificado por *smtp_ID*. No es obligatorio tener direcciones en el campo Cc:. Todas las direcciones que figuran en el área de encabezado "Para" y "cc" de un mensaje electrónico son visibles para cada destinatario del mensaje.

smtp_ID es la referencia entero largo de un mensaje creado con el comando *SMTP_New*.

copiaCarbon contiene una o más direcciones electrónicas.

eliminarOpcion es un valor entero que especifica si desea añadir o eliminar el encabezado "Cc":

- Un valor de cero, añade el nuevo valor al campo "Cc".
- Si pasa un valor de 1, el contenido del parámetro pasado reemplaza el contenido del encabezado existente. En este caso, si pasa una cadena vacía en *copiaCarbon*, el encabezado "Cc" se eliminará.
- Si pasa un valor de 2, el encabezado "Cc" se borra del mensaje.
  eliminarOpcion es un parámetro opcional que por defecto vale cero si no se especifica lo contrario.

Ejemplo

Ver el ejemplo del comando *SMTP_Body*.

SMTP_Charset

SMTP_Charset (codifEncab ; conjCuerpos) -> resultado

Parámetro	Tipo		Descripción
codifEncab	Entero	➡	-1 = Utilizar la configuración actual, 0 = No hacer nada, 1 = Convertir utilizando el conjunto de caracteres especificado si ISO-8859-1 o ISO-2022-JP, codificar los caracteres extendidos
conjCuerpos	Entero	➡	-1 = Utilizar el parámetro actual, 0 = No hace nada, 1 = Convertir en el conjunto de caracteres Mac OS si ISO-8859-1 o ISO-2022-JP
resultado	Entero	↩	Código de error

Descripción

El comando **SMTP_Charset** permite el soporte automático de los mensajes que contienen caracteres extendidos, durante su envío con los comandos *SMTP_QuickSend* o *SMTP_Send*.

El comando **SMTP_Charset** permite, por una parte, definir si el valor del parámetro ***SMTP_SetPrefs*** *caractYcodif* debe aplicarse para convertir los encabezados de los mensajes, nombres de archivos adjuntos y cuerpos, por otra parte, permite definir si un encabezado contiene caracteres extendidos que debe ser codificados utilizando la sintaxis "=?ISO-8859-1?Q?Test=E9?= …" como lo especifica la RFC1342. Este comando tiene un alcance interprocesos y tendrá efecto en todos los mensajes posteriores que se envíen con *SMTP_QuickSend* y *SMTP_Send* en todos los procesos 4D.

Este comando es especialmente útil para soportar caracteres extendidos incluidos en los encabezados del mensaje, tales como Asunto o los nombres insertados en las direcciones (por ejemplo, para la codificación de direcciones como "=?ISO-8859-1?Q?Test=E9?= <test@n.net >").

Los encabezados de los mensajes y los nombres de los archivos adjuntos serán codificados de la siguiente manera, de acuerdo a la RFC1342:

- Para los encabezados Asunto y Comentarios y los nombres de los archivos adjuntos: toda la cadena se codifica en base 64 si incluye caracteres extendidos.
- Para los encabezados From, To, CC, Bcc, Sender, ReplyTo, InReplyTo:
  - Cualquier texto entre ("<", ">") es sistemáticamente considerado como una dirección de correo electrónico y no se codifica.
  - Los caracteres separadores tales como SPC < > ( ) @ , ; : " / ? . = no se codifican.
  - Las cadenas delimitadas por caracteres especiales se codifican en base 64 si incluyen caracteres extendidos.
  - Ejemplos de direcciones:
    someone@somewhere no se codifica;
    Michèle <michele@somewhere>, sólo se codifica el nombre Michèle.
- Los otros encabezados no son codificados.

El parámetro *codifEncab* especifica cómo manejar la conversión de encabezado y la codificación durante el envío de un mensaje. El valor por defecto es 0.

- -1: Usar la configuración actual;
- 0: Valor por defecto: el conjunto de caracteres UTF-8 para el asunto, ISO-8859-1 para otros campos;
- 1: El conjunto de caracteres para los encabezados y nombres de archivos adjuntos se define por el parámetro *caractYcodif* **SMTP_SetPrefs**

Nota: los encabezados extendidos de tipo X_Mailer deben estar en ASCII US.

El parámetro *conjCuerpos* define cómo manejar el conjunto de caracteres de cuerpo de mensaje y la conversión de codificación y la codificación mientras se envía un mensaje. El valor por defecto es 0.

- -1: Usar la configuración actual;
- 0: Valor por defecto: UTF-8 base 64;
- 1: Utilizar el conjunto de caracteres y la codificación definida por el parámetro *caractYcodif* **SMTP_SetPrefs**

Nota: se recomienda utilizar la configuración predeterminada, que es apropiada para la mayoría de los sistemas/aplicaciones actuales.

Ejemplo

En este ejemplo, el asunto y el cuerpo se convierten utilizando el conjunto de caracteres UTF-8 y el asunto está codificado de acuerdo a la sintaxis RFC 1342:

```
SMTP_SetPrefs(1;1;0)
$err:=SMTP_Charset(1;1)
$err:=SMTP_QuickSend("mymail.com";"myaddress";"destination";"the Euro €";"the Euro symbol is €")
```

SMTP_Clear

SMTP_Clear (smtp_ID) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo		Referencia del mensaje
			0 si la ejecución es correcta
resultado	Entero		Código de error

Descripción

El comando *SMTP_Clear* borra el mensaje diseñado *smtp_ID*, liberando la memoria utilizada durante su creación. Cada llamada a *SMTP_New* debe tener una llamada correspondiente a *SMTP_Clear*.

smtp_ID contiene la identificación del mensaje electrónico creado con el comando *SMTP_New*.
Si el comando *SMTP_Clear* se ejecuta correctamente, *smtp_ID* devuelve el valor 0 (cero).

Ejemplo

Consulte los ejemplos de los comandos *SMTP_Body* y *SMTP_Send*.

SMTP_Comments

SMTP_Comments (smtp_ID ; comentarios {; eliminarOpcion}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
comentarios	Texto	➡	Texto del comentario
eliminarOpcion	Entero	➡	0 = Reemplazar (si comentarios no está vacío), 1 = Reemplazar, 2 = Borrar
resultado	Entero	↩	Código de error

Descripción

El comando *SMTP_Comments* permite al usuario añadir comentarios al mensaje sin modificar el cuerpo del mensaje. Los comentarios sólo aparecen en el área de encabezado del mensaje. Muchos softwares de correo no muestran el texto completo de los encabezados del mensaje a sus usuarios.

smtp_ID es la referencia entero largo de un mensaje electrónico creado con el comando *SMTP_New*.

comentarios contiene el texto a colocar como comentario en el encabezado del mensaje.

Atención: el texto no debe contener retornos de línea (ASCII = 10). Un retorno de línea significa el final de la sección de encabezado y el comienzo del cuerpo. Los encabezados posteriores podrían ser considerados como el cuerpo del texto y no ser reconocidos correctamente por el software del servidor o del cliente. Para obtener más información sobre los encabezados, consulte la RFC#822.

eliminarOpcion es un valor entero que especifica si se debe conservar o eliminar el encabezado "Comentarios":

- Si pasa cero, el contenido del parámetro *comentarios*, reemplaza en el área de encabezado todos texto presente, si pasa una cadena vacía en *comentarios*, se conserva el encabezado anterior.
- Si pasa 1, el contenido del parámetro *comentarios* reemplaza en el área de encabezado todo texto presente. En este caso, si pasa una cadena vacía en *comentarios*, se borra el encabezado "Comentarios".
- Si pasa un valor de 2, el encabezado "Comentarios" se borra del mensaje.
  eliminarOpcion es un parámetro opcional que por defecto tiene el valor cero.

Ejemplo

Ver el ejemplo del comando *SMTP_Body*.

SMTP_Date

SMTP_Date (smtp_ID ; msgFecha ; msgHora ; zonaHoraria ; offset {; eliminarOpcion}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
msgFecha	Fecha	➡	Fecha de creación del mensaje
msgHora	Hora	➡	Hora de creación del mensaje
zonaHoraria	Entero	➡	Código de ubicación
offset	Entero	➡	Depende del valor del parámetro zonaHoraria
eliminarOpcion	Entero	➡	0 = Añadir/Reemplazar, 1 = Borrar
resultado	Entero	↩	Código de error

Descripción

Dada una fecha, una hora, y una ubicación geográfica del creador del mensaje, el comando *SMTP_Date* crea el encabezado Date del mensaje especificado por *smtp_ID*. La fecha y hora que se pasan al comando deben corresponder a la ubicación actual de la máquina que envía el mensaje. Como los parámetros a continuación deben respetar un formato específico, de manera que el servidor de correo en el extremo receptor del mensaje puede interpretar la fecha y la hora locales según la fecha, la hora, la zona horaria y el offset pasados.

Nota: si un mensaje de correo electrónico está compuesto sin encabezado Date, el servidor SMTP añade uno en función de su configuración actual de fecha y hora. Todos los mensajes electrónicos SMTP contienen un encabezado Date, ya sea añadido por la aplicación cliente o por el servidor SMTP

smtp_ID es la referencia entero largo de un mensaje electrónico creado con el comando *SMTP_New*.

msgFecha es una fecha 4D que contiene la fecha en que este mensaje fue creado.

msgHora indica la hora de creación del mensaje.

zonaHoraria identifica la zona horaria del remitente. Este campo acepta un valor entre 0 y 6 en función de las siguientes indicaciones.

- Un valor de 0 (cero) permite al usuario especificar directamente en el parámetro offset el número de horas para sumar o restar de la hora universal.
- n valor de 1 hace que la máquina que envía añada automáticamente el offset basada en la PRAM del Macintosh. Si la zona horaria es 1 el parámetro offset no es necesario. La zona horaria de un ordenador Macintosh está determinada por las preferencias del sistema fecha y hora. Los desarrolladores deben tener en cuenta la exactitud de esta opción si los valores de tiempo son un factor crítico para sus bases de datos.
- Los valores de 2 a 5 corresponden a las cuatro zonas horarias en los Estados Unidos. El offset para cada uno de estos valores especifica si la zona horaria está en horario de verano (*offset* = 1) o no (*offset* = 0).
- Un valor de 6 especifica que la hora se da en hora militar. En este caso, el offset está determinado por la tabla horaria a continuación. Utilice el valor de offset correspondiente (-12 a 12), basado en el código horario en 24 horas de la localización del remitente.

El valor del parámetro *offset* depende del código definido en el parámetro *zonaHoraria.* Ver las descripciones anteriores o la tabla de abajo para encontrar el valor correcto a pasar en este parámetro.

Código	Zona Horaria	Parámetro Offset
0	+/- offset UT	Offset en horas +/-
1	+/- offset UT	Offset no utilizado, offset suministrado por la PRAM de Mac
2	EST - EDT	(0 = EST, 1 = EDT)
3	CST - CDT	(0 = CST, 1 = CDT)
4	MST - MDT	(0 = MST, 1 = MDT)
5	PST - PDT	(0 = PST, 1 = PDT)
6	Hora militar	Ver tabla a continuación

Valores de offset	**Códigos horarios militares**
0	Z
-1 a -9	A a I
-10 a -12	K a M
1 a 12	N a Y

Definiciones de las abreviaturas

UT	Hora Universal
EST	Hora Estándar del Este
EDT	Hora de verano del este
CST	Hora Central
CDT	Hora de verano Central
MST	Tiempo estándar de la Montaña
MDT	Hora de verano de las Montañas
PST	Hora Estándar del Pacífico
PDT	Hora de verano del Pacífico

eliminarOpcion: Un valor de cero agregará el encabezado Date con los parámetros dados, o reemplaza el anterior. Un valor de 1 elimina todo valor previamente definido en este encabezado. Los valores de los otros parámetros se ignoran. *eliminarOpcion* es un parámetro opcional que por defecto tiene el valor cero.

SMTP_Encrypted

SMTP_Encrypted (smtp_ID ; msgEncriptado {; eliminarOpcion}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
msgEncriptado	Texto	➡	Tipo de encriptación
eliminarOpcion	Entero	➡	0 = Reemplazar (si msgEncriptado no está vacío), 1 = Reemplazar, 2 = Borrar
resultado	Entero	⮌	Código de error

Descripción

El comando *SMTP_Encrypted* informa al usuario del tipo de encriptación utilizado en el cuerpo del mensaje. 4D Internet Commands **no** permite encriptar o descifrar mensajes electrónicos. El cifrado del cuerpo del mensaje se deja a los desarrolladores. Si el cuerpo del mensaje se encripta (antes de su asignación vía *SMTP_Body*), es necesario utilizar este comando para identificar el método de encriptación empleado.

smtp_ID es la referencia entero largo de un mensaje electrónico creado con el comando *SMTP_New*.

msgEncriptado es un valor texto que especifica el tipo de encriptación utilizado. El encabezado "Encriptado" es utilizado por el software de mensajería del destinatario para descifrar el cuerpo del mensaje recibido. Para más información sobre el formato de este parámetro, consulte la RFC # 822.

Atención: Tel texto no debe contener retornos de línea (ASCII = 10). Un retorno de línea significa el final de la sección de encabezado y el comienzo del cuerpo. Los encabezados posteriores podrían ser considerados como el cuerpo del texto y no ser reconocidos correctamente por el software del servidor o del cliente. Para obtener más información sobre los encabezados, consulte la RFC#822.

eliminarOpcion es un valor entero que especifica si se debe conservar o eliminar el encabezado "Encriptado":

- Si pasa cero, el contenido del parámetro msgEncriptado reemplaza en el área de encabezado todos texto presente, si pasa una cadena vacía en *msgEncriptado*, se conserva el encabezado anterior.
- Si pasa 1, el contenido del parámetro msgEncriptado reemplaza en el área de encabezado todo texto presente. En este caso, si pasa una cadena vacía en *msgEncriptado*, se borra el encabezado.
- Un valor de 2 eliminará el encabezado "Encriptado" del mensaje.
  eliminarOpcion es un parámetro opcional que por defecto tiene el valor cero.

SMTP_From

SMTP_From (smtp_ID ; msgDe {; eliminarOpcion}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
msgDe	Texto	➡	Dirección electrónica o Lista de direcciones
eliminarOpcion	Entero	➡	0 = Añadir, 1 = Reemplazar, 2 = Borrar
resultado	Entero	⮌	Código de error

Descripción

El comando *SMTP_From* contiene la(s) dirección(es) electrónicas de la(s) persona(s) en el encabezado "De" del mensaje. Las direcciones en este campo son de las personas responsables de crear o autorizar el mensaje. Generalmente, el encabezado "De" contiene la dirección de la persona que compuso y envió el mensaje. Sin embargo, cuando el mensaje es creado por un grupo de personas, cada miembro debe listarse individualmente en el encabezado "De".

El encabezado "De" es obligatorio. Si una dirección figura en el encabezado "De", la presencia del encabezado "Remitente" es opcional.

smtp_ID es la referencia entero largo de un mensaje electrónico creado con el comando *SMTP_New*.

msgRemitente es un valor texto que contiene una o más direcciones electrónicas. Todas las direcciones que figuran en el encabezado De son visibles para todos los destinatarios del mensaje.

Nota sobre las respuestas automáticas: si no se define un encabezado "ResponderA" para el mensaje identificado por *smtp_ID*, todas las respuestas al mensaje se dirigen a cada persona especificada en el encabezado "De".

eliminarOpcion es un valor entero que especifica si desea añadir o eliminar el campo "De" del encabezado:

- Un valor de cero, añade el nuevo valor al campo "De".
- Si pasa un valor de 1, el contenido del parámetro pasado reemplaza el contenido del encabezado existente. En este caso, si pasa una cadena vacía en *msgRemitente*, el encabezado "Remitente" se eliminará.
- Si pasa 2, el encabezado "Remitente" se borra del mensaje.
  eliminarOpcion es un parámetro opcional que por defecto tiene el valor cero.

Ejemplo

En este ejemplo, tres personas escriben un mensaje, sobre una modificación de la política de la empresa, que debe distribuirse a todo el personal de la empresa. Las respuestas a este mensaje serán direccionadas a las personas presentes en el encabezado "De".

```
$From:="prez@acme.com, vp@acme.com, cfo@acme.com"
$Error:=SMTP_From($smtp_id;$From;0)
$Error:=SMTP_Subject($smtp_id;"Cambio de política de la empresa";0)
$Error:=SMTP_To($smtp_id;◇Todos_Empleados;0)
```

SMTP_GetPrefs

SMTP_GetPrefs (retornoLinea ; caracYcodif ; longLinea) -> resultado

Parámetro	Tipo		Descripción
retornoLinea	Entero	⇐	0 = No añadir, 1 = Añadir
caracYcodif	Entero largo	⇐	Conjunto de caracteres del cuerpo del mensaje, encabezados y nombres de archivos adjuntos, así como también la codificación del cuerpo
longLinea	Entero largo	⇐	Longitud de línea máxima
resultado	Entero	⇒	Código de error

Descripción

El comando *SMTP_GetPrefs* devuelve los parámetros actuales de las preferencias SMTP. Los valores por defecto se devuelven en su estado por defecto a menos que una llamada previa a *SMTP_SetPrefs* altere la configuración. Para una descripción completa de estos parámetros, consulte el comando *SMTP_SetPrefs*.

El parámetro *retornoLinea* devuelve la configuración actual que determina cómo los comandos SMTP manejan los retornos de carro en el cuerpo de un mensaje.

caracYcodif devuelve la configuración actual para el cuerpo, encabezados y nombres de archivos adjuntos. Ver la tabla *caracYcodif* bajo *SMTP_SetPrefs* para una descripción de combinaciones correspondientes a valores.

El parámetro *longLinea* devuelve la longitud máxima actual de línea de texto en el cuerpo del mensaje.

SMTP_Host

SMTP_Host (smtp_ID ; nomServidor {; eliminarOpcion}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
nomServidor	Cadena	➡	Nombre o dirección IP del servidor
eliminarOpcion	Entero	➡	0 = Utilizar o reemplazar, 1 = Borrar
resultado	Entero	⮌	Código de error

Descripción

Todos los mensajes creados y enviados con los comandos SMTP deben dirigirse a un servidor SMTP determinado. 4D Internet Commands no distribuye el correo directamente a cada destinatario, sino al servidor SMTP especificado por este comando. El servidor SMTP es responsable de resolver los posibles errores de dirección y de la distribución del mensaje.

smtp_ID es la referencia entero largo de un mensaje electrónico creado con el comando *SMTP_New*.

nomServidor es el nombre o la dirección IP del servidor de correo SMTP que se encargará de la distribución del mensaje.

eliminarOpcion es un parámetro opcional que especifica si desea eliminar la configuración del servidor actual. Un valor de cero hace que el servidor utilice el valor especificado por *nomServidor*. Un valor de 1, eliminará la especificación del servidor para el mensaje identificado por *smtp_ID*. Este es un parámetro opcional que por defecto vale cero si no se especifica lo contrario.

Ejemplo

Ver los ejemplos de los comandos *SMTP_Body* y *SMTP_Send*.

SMTP_InReplyTo

SMTP_InReplyTo (smtp_ID ; msgEnRespuestaA {; eliminarOpcion}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
msgEnRespuestaA	Texto	➡	Texto en respuesta a
eliminarOpcion	Entero	➡	0 = Reemplazar (si msgEnRespuetaA no está vacío), 1 = Reemplazar, 2 = Borrar
resultado	Entero	↩	Código de error

Descripción

El comando **SMTP_InReplyTo** inserta, al responder a un mensaje, el texto del mensaje original.

smtp_ID es la referencia entero largo de un mensaje electrónico creado con el comando *SMTP_New*.

msgEnRespuestaA contiene el texto de los mensajes anteriores al que pertenece este mensaje. Para más información sobre el formato de este parámetro, consulte la RFC # 822.

Atención: el texto no debe contener retornos de línea (ASCII = 10). Un retorno de línea significa el final de la sección de encabezado y el comienzo del cuerpo. Los encabezados posteriores podrían ser considerados como el cuerpo del texto y no ser reconocidos correctamente por el software del servidor o del cliente. Para obtener más información sobre los encabezados, consulte la RFC#822.

eliminarOpcion es un valor entero que especifica si se debe conservar o eliminar el encabezado "Responder a":

- Si pasa cero, el contenido del parámetro responderA reemplaza en el área de encabezado todo texto presente, si pasa una cadena vacía en *msgEnRespuestaA*, se conserva el encabezado anterior.
- Si pasa 1, el contenido del parámetro responderA reemplaza en el área de encabezado todo texto presente. En este caso, si pasa una cadena vacía en *msgEnRespuestaA*, se borra el encabezado "*responderA*" existente).
- Si pasa un valor de 2, el encabezado "responderA" se borra del mensaje.
eliminarOpcion es un parámetro opcional que por defecto tiene el valor cero.

SMTP_Keywords

SMTP_Keywords (smtp_ID ; msgPalabrasClaves {; eliminarOpcion}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
msgPalabrasClaves	Texto	➡	Lista de palabras claves
eliminarOpcion	Entero	➡	0 = Reemplazar (si msgPalabrasClaves no está vacío), 1 = Reemplazar, 2 = Borrar
resultado	Entero	↩	Código de error

Descripción

El comando *SMTP_Keywords* se utiliza para insertar las palabras clave en el encabezado "palabras claves" del mensaje designado por *smtp_ID*.

smtp_ID es la referencia entero largo de un mensaje electrónico creado con el comando *SMTP_New*.

msgPalabrasClaves es un valor texto que contiene una o varias palabras claves, separadas por barras diagonales. Para más información sobre el formato de este parámetro, consulte la RFC#822.

Atención: el texto no debe contener retornos de línea (ASCII = 10). Un retorno de línea significa el final de la sección de encabezado y el comienzo del cuerpo. Los encabezados posteriores podrían ser considerados como el cuerpo del texto y no ser reconocidos correctamente por el software del servidor o del cliente. Para obtener más información sobre los encabezados, consulte la RFC#822.

eliminarOpcion es un valor entero que especifica si se debe conservar o eliminar el encabezado "palabras claves":

- Si pasa cero, el contenido del parámetro msgPalabrasClaves reemplaza en el área de encabezado todo texto presente, si pasa una cadena vacía en *msgPalabrasClaves*, se conserva el encabezado anterior.
- Si pasa 1, el contenido del parámetro msgPalabrasClaves reemplaza en el área de encabezado todo texto presente. En este caso, si pasa una cadena vacía en *msgPalabrasClaves*, el encabezado "Palabras claves" se borra del mensaje.
- Si pasa un valor de 2, el encabezado "Palabras claves" se borra del mensaje.
  eliminarOpcion es un parámetro opcional que por defecto tiene el valor cero.

SMTP_MessageID

SMTP_MessageID (smtp_ID ; mensaje_ID {; opcion}) -> Resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
mensaje_ID	Texto	➡	Identificación única del mensaje
opcion	Entero	➡	0 = Añadir (predeterminado), 1 = Reemplazar, 2 = Suprimir
Resultado	Entero	⮌	Código de error

Descripción

El comando **SMTP_MessageID** añade un campo "mensaje-ID " en el encabezado del mensaje cuya referencia se pasa en *smtp_ID*. Este identificador único se utiliza en particular en los foros o listas de correo públicas. En general, los servidores de correo añaden automáticamente este encabezado a los mensajes que envían. Puede utilizar este comando para definir su contenido.

smtp_ID contiene el ID de un mensaje de correo electrónico creado con el comando **SMTP_New**.

En *mensaje_ID*, pase el identificador a asociar al mensaje. Los contenidos a pasar son normalmente sin restricciones, sin embargo por convención, por lo general son de la forma "*letrasONumeros@dominio*", por ejemplo "abcdef.123456@4d.com". Tenga en cuenta que algunos servidores de correo electrónico (como Gmail) no reconocen los encabezados "message-id" personalizados y los reemplazan cuando no están en esta forma.

El parámetro *opcion* le permite especificar si conservar o eliminar el encabezado *mensaje_ID* existente:

- Si pasa 0 (valor por defecto si el parámetro se omite), se añade el contenido del parámetro pasado al contenido existente.
- Si pasa 1, el contenido del parámetro pasado reemplaza el contenido existente.
- Si pasa 2, los contenidos existentes se eliminan del mensaje.

Ejemplo

En este ejemplo, un mensaje con un encabezado "message-id" específico se envía para cada registro de la tabla [Admin]:

```
$error:=SMTP_New($smtp_id)
$error:=SMTP_Host($smtp_id;"infoserv.com")
$error:=SMTP_From($smtp_id;"info@infoserv.com")
$error:=SMTP_Subject($smtp_id;"General statistics")
FIRST RECORD([Admin])
For($i;1;Records in selection([Admin]))
   $error:=SMTP_Body($smtp_id;$Stats)
   $error:=SMTP_To($smtp_id;[Admin]Email;1)  // Remplaza el encabezado "A" por un nuevo valor
   $error:=SMTP_MessageID($smtp_id;[Admin]ID+"@infoserv.com";1)  // Uso del  ID del admin
   $error:=SMTP_Send($smtp_id)
   NEXT RECORD([Admin])
End for
$error:=SMTP_Clear($smtp_id)
```

SMTP_New

SMTP_New (smtp_ID) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	⇐	Referencia del nuevo mensaje
resultado	Entero	⇒	Código de error

Descripción

El comando *SMTP_New* debe ser el primer comando llamado en cualquier secuencia de creación de un mensaje de correo electrónico SMTP, excepto si *SMTP_QuickSend* se está utilizando. *SMTP_New* crea un nuevo mensaje en memoria y devuelve una referencia entero largo en el parámetro *smtp_ID*. Los comandos SMTP posteriores utilizarán la referencia *smtp_ID* para llenar la información del encabezado y del cuerpo del mensaje antes de llamar a *SMTP_Send*.

Cada llamada a *SMTP_New* debe tener una llamada correspondiente a *SMTP_Clear*. Después de enviar un mensaje, la llamada a *SMTP_Clear* libera la memoria ocupada por el contenido del mensaje.

smtp_ID es la referencia entero largo al mensaje que acaba de crear. Esta identificación es utilizada por todas las referencias posteriores a este mensaje. Es posible abrir simultáneamente varios nuevos mensajes y el *smtp_ID* devuelto para cada uno ofrece un medio para identificar cuál es el mensaje al que se debe aplicar un comando.

Ejemplos

Ver los ejemplos de los comandos *SMTP_Body* y *SMTP_Send*.

SMTP_QuickSend

SMTP_QuickSend (nomServidor ; msgDe ; msgA ; msgObjeto ; mensaje {; paramSesion}{; puerto}{; nomUsuario ; contrasena}) -> resultado

Parámetro	Tipo		Descripción
nomServidor	Cadena	➡	Nombre o dirección IP del servidor
msgDe	Texto	➡	Dirección electrónica o Lista de direcciones
msgA	Texto	➡	Dirección electrónica o Lista de direcciones
msgObjeto	Texto	➡	Asunto del mensaje (UTF-8 por defecto)
mensaje	Texto	➡	Mensaje (UTF-8 por defecto)
paramSesion	Entero largo	➡	0 o se omite = No utilizar SSL sino switchover, 1 = Utilizar SSL, 2 = Nunca utilizar SSL (switchover no permitido), 4 = Enviar texto HTML sin SSL, 5 = Enviar texto HTML con SSL, 8 = Enviar Mime HTML sin SSL/TLS, 9 = Enviar Mime HTML con SSL/TLS
puerto	Entero largo	➡	Número de puerto a utilizar
nomUsuario	Texto	➡	Nombre de usuario para la autenticación
contrasena	Texto	➡	Contraseña para la autenticación
resultado	Entero	↩	Código de error

Descripción

El comando **SMTP_QuickSend** permite crear y enviar un mensaje con un solo comando. Si requiere un mayor control sobre el mensaje o si el mensaje es más complejo, utilice el comando **SMTP_New**.

Nota: este comando no se puede utilizar en las bases convertidas que funcionan en modo "no Unicode".

nomServidor contiene el nombre o la dirección IP del servidor SMTP que se encargará de la distribución del mensaje.

msgDe contiene una o más direcciones electrónicas completas indicando quien envió el mensaje originalmente. Todas las direcciones que figuran en el encabezado De son visibles para todos los destinatarios del mensaje.

msgA contiene una o más direcciones electrónicas completas. Las direcciones identificadas en el encabezado msgA reciben una copia original del mensaje. Cada destinatario del mensaje puede ver las otras direcciones electrónicas a las cuales fue enviado el mensaje.

asunto es un valor texto que describe de forma concisa el tema tratado en detalle por el cuerpo del mensaje.

Nota: por defecto, el asunto y el cuerpo del mensaje están codificados en UTF-8, lo que garantiza que los caracteres enviados serán interpretados correctamente por casi todos los clientes de correo electrónico. Si desea utilizar un conjunto de caracteres específicos, consulte los comandos **SMTP_SetPrefs** y **SMTP_Charset**.

mensaje es un valor texto que contiene el cuerpo del mensaje.

El parámetro opcional *paramSesion* establece el formato del mensaje (texto estándar, HTML o Mime HTML) y el modo de activación del protocolo SSL para la conexión:

- Si pasa 0 u omite este parámetro, el mensaje será formateado en texto y enviado en modo estándar no seguro. Si el servidor propone una actualización a SSL/TLS después de la autenticación, la báscula se efectúa automáticamente (funcionamiento del SSL/TLS en modo explícito).
- Si pasa 1, el mensaje será formateado en texto y enviado enn SSL (modo síncrono),
- Si pasa 2, el mensaje será formateado en texto y enviado pero sin soporte de actualización en SSL/TLS.
- Si pasa 4, el mensaje será formateado en HTML y enviado en modo estándar.
- Si pasa 5, el mensaje será formateado en HTML y enviado en modo SSL/TLS,
- Si pasa 8, el mensaje será formateado en Mime HTML y enviado en modo estándar,
- Si pasa 9, el mensaje será formateado en Mime HTML y enviado en modo SSL/TLS,
  Nota: Mime HTML (extensión de archivo .mht o .mhtml) es un formato de archivo de página Web que pueden fusionar el código HTML, así como también los recursos externos, como imágenes en un único documento. Se apoya en varios navegadores, así como MS Word, por ejemplo. Dado que este formato es soportado para áreas 4D Write Pro, puede fácilmente guardar y enviar áreas 4D Write Pro por correo electrónico incluyendo todos sus recursos.

Tenga en cuenta que estos valores corresponden a las combinaciones habituales, sin embargo el parámetro *paramSesion* es un realidad un campo de *bits* y permite cualquier combinación personalizada si utiliza **Operadores de bits**:

Número de bit	Formato utilizado si el bit es 1
0	Usar SSL o el comportamiento predeterminado, conexión en claro y actualizar a SSL si es posible.
1	Nunca actualizar, permanecer en modo no cifrado, incluso si la actualización es posible. Bit se ignora si se ha seleccionado SSL (bit-0).
2	Cuerpo del mensaje en HTML, definir el encabezado apropiado.
3	Mensaje MHTML, el bit 2 (HTML) se ignora. El usuario es responsable de configurar todo, excepto "A", "De", "Fecha" y "Asunto

El parámetro opcional *puerto* especifica el número de puerto SMTP a utilizar para la conexión con el servidor. Los valores utilizados con más frecuencia son:

- 25 = puerto STMP estándar no seguro (puerto por defecto si el parámetro se omite)
- 465 = puerto SMTPS (SSL/TLS)
- 587 = puerto STMP estándar (seguro); pase este puerto para conexiones con un servidor MS Exchange (modo explícito).

Los parámetros opcionales *nomUsuario* y *contrasena* se utilizan para autenticar el remitente con el servidor de correo. Estos parámetros deben pasarse en conjunto. Note que el modo de autenticación más seguro soportado pro el servidor será el utilizado (como el modo por defecto del comando **SMTP_Auth**).

Ejemplo 1

Ejemplo de uso de este comando:

```
$Host:="www.4d.com"
$ToAddress:="adupont@4d.fr"
$FromAddress:="jsmith@4d.com"
$Subject:="Informes de ventas"
$Message:="¿Podría enviarme el informe de ventas de enero de 2009? Gracias."
$Error:=SMTP_QuickSend($Host;$FromAddress;$ToAddress;$Subject;$Message;1)
If($Error#0)
   ALERT("Error: SMTP_QuickSend"+Char(13)+IT_ErrorText($Error))
End If
```

Ejemplo 2

Ejemplo de utilización del comando para un envío de mensaje seguro vía un servidor MS Exchange:

```
$ServerName:="exchange.4d.com"
$MsgTo:="adupont@gmail.com"
$MsgFrom:="a.user@4d.com"
$Subject:="Test message"
$Message:="This is a test for sending a message in secure mode. Please do not reply."
$Error:=SMTP_QuickSend($ServerName;$MsgFrom;$MsgTo;$Subject;$Message;0;587;"a.user";"@!password@!")
```

Ejemplo 3

Envío de un mensaje en HTML con SSL/TLS:

```
$Host:="smtp.gmail.com"
$ToAddress:="john@4d.com"
$FromAddress:="harry@gmail.com"
$Subject:="Message HTML"
$Message:="Let's meet at <b>Joe's Coffee Shop</b>!"
$Param:=5  //HTML with SSL
$Port:=465  //SSL port of gmail
$User:="harry@gmail.com"
$Password:="xyz&@!&@"
$Error:=SMTP_QuickSend($Host;$FromAddress;$ToAddress;$Subject;$Message;$Param;$Port;$User;$Password)
```

Ejemplo 4

Usted guardó un documento .mht de su disco y desea enviarlo por correo electrónico. Para ello, puede escribir:

```
$Message:=Document to text("c:\\documents\\invitation.mht")
$Host:="smtp.gmail.com"
$ToAddress:="john@4d.com"
$FromAddress:="harry@gmail.com"
$Subject:="Let's party"
$Param:=9  //MHTML with SSL
$Port:=465  //SSL port of gmail
$User:="harry@gmail.com"
$Password:="xyz&@!&@"
$Error:=SMTP_QuickSend($Host;$FromAddress;$ToAddress;$Subject;$Message;$Param;$Port;$User;$Password)
```

SMTP_References

SMTP_References (smtp_ID ; msgReferencias {; eliminarOpcion}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
msgReferencias	Texto	➡	Texto de referencia
eliminarOpcion	Entero	➡	0 = Reemplazar (si msgReferencias no está vacío), 1 = Reemplazar, 2 = Borrar
resultado	Entero	⮌	Código de error

Descripción

El comando *SMTP_References* identifica la correspondencia adicional que hace referencia el mensaje.

smtp_ID es la referencia entero largo de un mensaje electrónico creado con el comando *SMTP_New*.

referencias es un valor de texto que contiene el texto de referencia. Para más información sobre el formato de este parámetro, consulte la RFC # 822.

Atención: el texto no debe contener retornos de línea (ASCII = 10). Un retorno de línea significa el final de la sección de encabezado y el comienzo del cuerpo. Los encabezados posteriores podrían ser considerados como el cuerpo del texto y no ser reconocidos correctamente por el software del servidor o del cliente. Para obtener más información sobre los encabezados, consulte la RFC#822.

eliminarOpcion es un valor entero que especifica si se debe conservar o eliminar el encabezado "Referencias":

- Si pasa cero, el contenido del parámetro Referencias reemplaza en el área de encabezado todos texto presente, si pasa una cadena vacía en *referencias*, se conserva el encabezado anterior.
- Si pasa 1, el contenido del parámetro Referencias reemplaza en el área de encabezado todo texto presente. En este caso, si pasa una cadena vacía en *referencias*, se borra el encabezado).
- Si pasa un valor de 2, el encabezado "Referencias" se borra del mensaje.
  eliminarOpcion es un parámetro opcional que por defecto tiene el valor cero.

SMTP_ReplyTo

SMTP_ReplyTo (smtp_ID ; replyTo {; eliminarOpcion}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	→	Referencia del mensaje
replyTo	Texto	→	Dirección electrónica o Lista de direcciones
eliminarOpcion	Entero	→	0 = Añadir a la lista existente, 1 = Reemplazar los antiguos valores para los nuevos, 2 = Eliminar las direcciones especificas
resultado	Entero	↩	Código de error

Descripción

El comando *SMTP_ReplyTo* permite al usuario controlar la dirección de las respuestas al mensaje. Normalmente, se utiliza el contenido del encabezado "De" para las respuestas a un mensaje. Sin embargo, el encabezado "De" se ignora cuando n encabezado "ReplyTo" se define para el mensaje.

Para el desarrollador de bases de datos, *SMTP_ReplyTo* puede ser una herramienta muy potente que les permite controlar el comportamiento de las respuestas en los mensajes automáticos. Los usuarios pueden querer respuestas enviadas a direcciones distintas de las listadas en los encabezados De o Para, como una cuenta separada creada para hacer seguimiento de las respuestas.

smtp_ID es la referencia entero largo al mensaje electrónico creado con el comando *SMTP_New*

respuestaA es un valor texto que contiene una o varias direcciones electrónicas. Las direcciones listadas en este encabezado serán utilizadas por el software de correo de los destinatarios como direcciones de respuesta por defecto.

eliminarOpcion es un valor entero que especifica cómo manejar la o las dirección(es) en *respuestaA*. Un valor de cero a añadir añade los nuevos valores a los asignados previamente a este encabezado. Un valor de 1 sustituirá las definiciones previas con los nuevos valores. Si *respuestaA* es una cadena vacía, todos los valores anteriores se eliminarán y el encabezado se borrará del mensaje. Un valor de 2, eliminará las direcciones especificadas de los valores previamente asignados. *eliminarOpcion* es un parámetro opcional que por defecto tiene el valor cero.

Ejemplo

En este ejemplo, 3 ejecutivos escriben un mensaje sobre un cambio de política de la empresa. Este mensaje debe ser enviado a todos los empleados por la secretaria. Las respuestas a este mensaje serán devueltas a la secretaria y al "personal_dept" y no a los ejecutivos.

```
$From:="prez@acme.com, vp@acme.com, cfo@acme.com"
$Error:=SMTP_From($smtp_id;$From;0)
$Error:=SMTP_Sender($smtp_id;"secretaria@acme.com";0)
$Error:=SMTP_ReplyTo($smtp_id;"secretaria@acme.com, personal_dept@acme.com";0)
$Error:=SMTP_Subject($smtp_id;"Cambio en política de la empresa";0)
$Error:=SMTP_To($smtp_id;◇Todos_Empleados;0)
```

SMTP_Send

SMTP_Send (smtp_ID {; paramSesion}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
paramSesion	Entero largo	➡	0 o se omite = No utilizar SSL sino báscula permitida, 1 = Utilizar SSL, 2 = Nunca utilizar SSL (báscula no permitida)
resultado	Entero	⮌	Código de error

Descripción

El comando *SMTP_Send* envía el mensaje referenciado por *smtp_ID* pero no borra los datos de la memoria.

smtp_ID es la referencia entero largo de un mensaje electrónico creado con el comando *SMTP_New*.

El parámetro opcional *sessionParam* permite activar el protocolo SSL para la conexión:

- Si pasa 0 u omite este parámetro, el mensaje se enviará en modo estándar no seguro. Si el servidor ofrece una actualización a SSL/TLS luego de la actualización, la báscula se realiza automáticamente (funcionamiento SSL/TLS en modo explícito).
- Si pasa 1, el envío del mensaje se efectúa en SSL (modo síncrono),
- Si pasa 2, el envío del mensaje se efectúa en modo estándar pero sin soporte de la actualización en SSL/TLS.

Notas sobre el uso de STARTTLS (modo explícito)

A partir de la versión 13.2, 4D Internet Commands STARTTLS en modo explícito. Esto significa que la conexión se hace primero en modo estándar y luego se "actualiza" en SSL/TLS después de la fase de autenticación. Consulte el ejemplo 2 para una ilustración de este mecanismo.

- La conexión inicial se debe hacer en un puerto no SSL/TLS que no es el puerto por defecto (25). Debe llamar al comando **IT_SetPort** antes de **SMTP_Send** para designar el puerto utilizado para la conexión SMTP inicial. Para una conexión a un servidor MS Exchange, debe utilizar el puerto 587.

- La conexión debe autenticarse por lo que tiene que llamar al comando **SMTP_Auth** . Sólo el modo de autenticación LOGIN es soportado por 4D Internet Commands para comunicarse con un servidor MS Exchange. Puede pasar este modo o dejar el modo por defecto (en este caso se utiliza el modo más seguro disponible en el servidor):
[# code4D]$error:=SMTP_Auth ($smtp_id"; user.name","password", 2)// OK para el modo LOGIN v$error:=SMTP_Auth ($smtp_id;"user.name";"password") // OK para el modo LOGIN definido por el servidor[#/code4D]

Ejemplo 1

En este ejemplo, se crea un mensaje y se definen los elementos estáticos. Para cada registro de la tabla [Personas], el mensaje se personaliza y se envía.

```
$error:=SMTP_New($smtp_id)
$error:=SMTP_Host($smtp_id;"wkrp.com")
$error:=SMTP_From($smtp_id;"herb_tarlick@wkrp.com")
$error:=SMTP_ReplyTo($smtp_id;"bigguy@wkrp.com")
$error:=SMTP_Subject($smtp_id;"¡Promociones en espacios publicitarios!")
FIRST RECORD([Personas])
For($i;1;Records in selection([Personas]))
   If([Personas]Sales2Date>100000)
      $Body:=◇BigDiscText
   Else
```

```
        $Body:=◇SmlDiscText
    End if
    $Body:=Replace string($BoilerPlate;"<Salutation>";[Personas]Firstname)
    $error:=SMTP_To($smtp_id;[Personas]Email;1)  `Remplazar el encabezado "A" por un nuevo valor
    $error:=SMTP_Body($smtp_id;$Body)
    $error:=SMTP_Send($smtp_id)
    NEXT RECORD([Personas])
End for
$error:=SMTP_Clear($smtp_id)
```

Ejemplo 2

Este ejemplo envía un mensaje de prueba vía un servidor Exchange en STARTTLS:

```
$error:=SMTP_New($smtp_id)
$error:=SMTP_Host($smtp_id;"exchange.4d.com")
$error:=SMTP_From($smtp_id;"username@4d.com")
$error:=SMTP_ReplyTo($smtp_id;"username@4d.com")
$error:=SMTP_Subject($smtp_id;"Message test")
$error:=SMTP_Auth($smtp_id;"username";"!%@password")  //utilizar identificadores válidos
$Body:="Esta es una prueba de envío de mensajes vía Exchange, favor no responder"
$error:=IT_SetPort(2;587)  //modo STMP estándar, pueto 587 para Exchange
$error:=SMTP_To($smtp_id;"recipient@gmail.com")
$error:=SMTP_Body($smtp_id;$Body)
$error:=SMTP_Send($smtp_id;0)  //Envío en modo 'actualizable'
ALERT(String($error));
```

SMTP_Sender

SMTP_Sender (smtp_ID ; msgRemitente {; eliminarOpcion}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
msgRemitente	Texto	➡	Dirección electrónica (1 únicamente)
eliminarOpcion	Entero	➡	0 = Añadir, 1 = Reemplazar, 2 = Borrar
resultado	Entero	⮌	Código de error

Descripción

El comando *SMTP_Sender* añade la dirección de correo electrónico de la persona que envía el mensaje. Este comando está destinado a ser utilizado cuando el remitente no es el autor real del mensaje, o para indicar entre un grupo de autores quien envió realmente el mensaje. Este comando no es necesario si el contenido del campo "Remitente" es redundante con el del campo "De".

Cuando los mensajes son creados y enviados automáticamente por un programa, el área de encabezado Remitente debe hacer referencia a la cuenta de correo del administrador del programa y no a la cuenta administrada por el programa.

smtp_ID es la referencia entero largo de un mensaje electrónico creado con el comando *SMTP_New*.

msgRemitente contiene la dirección electrónica a insertar en el encabezado Remitente del mensaje. Sólo una dirección de correo electrónico puede especificarse para este encabezado.

eliminarOpcion es un valor entero que especifica si desea añadir o eliminar el encabezado Remitente:

- Un valor de cero, añade el nuevo valor al campo Remitente.
- Si pasa un valor de 1, el contenido del parámetro pasado reemplaza el contenido del encabezado existente. En este caso, si pasa una cadena vacía en *msgRemitente*, e eliminará el encabezado "Remitente".
- Si pasa 2, el encabezado "Remitente" se borra del mensaje.
  eliminarOpcion es un parámetro opcional que por defecto tiene el valor cero.

Ejemplo

En este ejemplo, tres personas escriben un mensaje, sobre una modificación de la política de la empresa, que debe ser distribuido a todo el personal de la empresa por la secretaria. Las respuestas a este mensaje deben enviarse a las tres personas presentes en el encabezado "De".

```
$From:="prez@acme.com, vp@acme.com, cfo@acme.com"
$Error:=SMTP_From($smtp_id;$From;0)
$Error:=SMTP_Sender($smtp_id;"secretary@acme.com";0)
$Error:=SMTP_Subject($smtp_id;"Cambio de política de la empresa";0)
$Error:=SMTP_To($smtp_id;◇Todos_Empleados;0)
```

SMTP_SetPrefs

SMTP_SetPrefs (retornoLinea ; caracYcodif ; longLinea) -> resultado

Parámetro	Tipo		Descripción
retornoLinea	Entero	➡	1 = [por defecto] Añadir, 0 = No añadir, -1 = No cambiar
caracYcodif	Entero largo	➡	Conjunto de caracteres del cuerpo del mensaje, encabezados y nombres de archivos adjuntos, así como también codificación del cuerpo (-1 = sin cambio)
longLinea	Entero largo	➡	Longitud de línea máxima (0 = [por defecto] Detección auto, -1 = Ninguna modificación)
resultado	Entero	↩	Código de error

Descripción

El comando *SMTP_SetPrefs* define las preferencias de los mensajes a enviar utilizando los comandos SMTP. El comando tiene un alcance global e interproceso y afecta a todos los mensajes posteriores creados con los comandos SMTP. Las opciones configurables afectan el formato de un mensaje que se envía a un servidor SMTP con los comandos *SMTP_QuickSend* o *SMTP_Send*. La configuración de las preferencias tiene un alcance interprocesos y afecta la creación de correos en cualquier proceso 4D.

Los servidores SMTP requieren la combinación de caracteres retorno de carro/retorno de línea (CR/LF) para indicar el final de una línea. Esto difiere de la mayoría de las aplicaciones Mac, que consideran un retorno de carro como un marcador de fin de línea o de párrafo.

retornoLinea es un valor entero que especifica cómo manejar los retornos de carro en el cuerpo de un mensaje. Al pasar un valor de cero en este parámetro se deja el texto del cuerpo del mensaje intacto, lo que permite al desarrollador controlar sus propias adiciones de retornos de línea. Un valor de 1 (parámetro por defecto), reemplaza automáticamente todos los retorno de carro/retorno línea por retornos de carro solos. Un valor de -1 deja el valor actual de la preferencia intacto. Si no está seguro de qué opción elegir, debe elegir 1, el valor por defecto.

caracYcodif especifica el conjunto de caracteres utilizado en el cuerpo del mensaje, encabezados y nombres de archivos adjuntos a enviar, así como también la codificación a aplicar al cuerpo del mensaje, de acuerdo con los valores en la tabla abajo. Por ejemplo, “US-ASCII & 7 bit” (valor 2) significa que el conjunto de caracteres utilizado es el ASCII US, incluye sólo códigos ASCII estándar (0 a 127) que son comunes para Windows y Mac y 4D Internet Commands codificarán el cuerpo del mensaje utilizando la codificación de 7 bits. Tenga en cuenta que el comando *SMTP_SetPrefs* no convierte el cuerpo del mensaje utilizando el conjunto caracteres especificado, el usuario debe asegurarse de la conformidad del conjunto de caracteres. Si desea convertir el conjunto de caracteres utilizado en un mensaje, consulte la descripción del comando *SMTP_Charset*.
Si no cambia, el tipo de contenido por defecto es 1.

-1	Ninguna modificación	
0	Aplicación y binario; sin codificación	ISO-8859-1
1	Por defecto; elegirá "US-ASCII & 7bit" o "ISO-8859-1 & quotable-printable" basado en el contenido del mensaje.	
2	US-ASCII y 7bits	
3	US-ASCII y quotable-printable	
4	US-ASCII y base64	
5	ISO-8859-1 y quotable-printable	
6	ISO-8859-1 y base64	
7	ISO-8859-1 y 8 bits	
8	ISO-8859-1 y binario	
9	Reservado	
10	ISO-2022-JP (Japonés) y 7 bits	
11	ISO-2022-KR (Coreano) y 7 bits	
12	ISO-2022-CN (Tradicional y Chino simplificado) y 7 bits	
13	HZ-GB-2312 (Chino simplificado) y 7 bits	
14	Shift-JIS (Japonés) y base64	
15	UTF-8 y quoted-printable	
16	UTF-8 y base64	

Atención: el carácter € ("euro") no es parte de ISO-8859-1.

El parámetro *longLinea* especifica la longitud máxima de línea en el cuerpo del mensaje. Los comandos SMTP "fuerzan" el paso a la línea en el cuerpo del texto mediante la inserción de un retorno de carro/salto de línea después de la palabra más cercana de la longitud máxima de línea. Puede especificarse cualquier número, pero es preferible que las líneas no pasen de 80 caracteres. Un valor de -1 deja el valor actual intacto.
El valor por defecto del parámetro *longLinea* es cero. Un valor de cero hará que los comandos SMTP utilicen los valores recomendados especificados en las definiciones RFC para *conjuntoYcodificacion*. Si el parámetro *longLinea* vale cero, el ajuste se realizará con base en la siguiente tabla:

Tipo de cuerpo	**Pasa a la línea**
Base64	76
Quoted-Printable	76
Otros…	no pasar a la línea

El ajuste de línea es muy recomendable ya que muchos sistemas y programas de correo tienen problemas para manejar mensajes con líneas de longitud ilimitada. Además, tenga en cuenta que el correo a menudo viaja a través de una serie de sistemas antes de llegar a su destino final y cualquier equipo a lo largo de la ruta de entrega puede rechazar un mensaje si no es capaz de manejar el formato del mensaje.

Ejemplo

El siguiente código envía un mensaje en UTF-8 codificado en quotedprintable (los encabezados permanecen en su conjunto de caracteres por defecto):

```
$err:=SMTP_SetPrefs(-1;15;-1)
$err:=SMTP_Charset(0;1)  //aplicar las preferencias
$err:=SMTP_QuickSend("mymail.com";"myaddress";"destination";"El Euro €";"El símbolo Euro es €")
```

SMTP_Subject

SMTP_Subject (smtp_ID ; msgObjeto {; eliminarOpcion}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
msgObjeto	Texto	➡	Asunto del mensaje
eliminarOpcion	Entero	➡	0 = Reemplazar (si asunto no está vacío), 1 = Reemplazar, 2 = Borrar
resultado	Entero	⮌	Código de error

Descripción

El comando **SMTP_Subject** añade el asunto del mensaje al mensaje referenciado por *smtp_ID*. Si un asunto ya ha sido añadido por un comando *SMTP_Subject*, el nuevo asunto anulará el asunto anterior.

smtp_ID es la referencia entero largo de un mensaje electrónico creado con el comando *SMTP_New*.

asunto es un valor texto que describen de forma concisa el tema tratado en detalle por el cuerpo del mensaje.

Notas:

- Por defecto, el encabezado "Asunto" del mensaje está codificado en UTF-8, lo que garantiza que los caracteres enviados serán interpretados correctamente por casi todos los clientes de correo electrónico. Si desea utilizar un conjunto de caracteres específicos, consulte los comandos **SMTP_SetPrefs** y **SMTP_Charset**.
- El texto no debe contener retornos de línea (ASCII = 10). Un retorno de línea significa el final de la sección de encabezado y el comienzo del cuerpo. Los encabezados posteriores podrían ser considerados como el cuerpo del texto y no ser reconocidos correctamente por el software del servidor o del cliente. Para obtener más información sobre los encabezados, consulte la RFC#822.

eliminarOpcion es un valor entero que especifica si se debe conservar o eliminar el encabezado "Asunto":

- Si pasa cero, el contenido del parámetro asunto reemplaza en el área de encabezado todo texto presente, si pasa una cadena vacía en *asunto*,se conserva el encabezado anterior.
- Si pasa 1, el contenido del parámetro asunto reemplaza en el área de encabezado todo texto presente. En este caso, si pasa una cadena vacía en *asunto*, se borra el encabezado "asunto" existente.
- Un valor de 2 eliminará el encabezado "Asunto" del mensaje.
  eliminarOpcion es un parámetro opcional que por defecto tiene el valor cero.

Ejemplo

Ver el ejemplo del comando *SMTP_Body*.

SMTP_To

SMTP_To (smtp_ID ; msgA {; eliminarOpcion}) -> resultado

Parámetro	Tipo		Descripción
smtp_ID	Entero largo	➡	Referencia del mensaje
msgA	Texto	➡	Objeto del mensaje
eliminarOpcion	Entero	➡	0 = Añadir, 1 = Reemplazar, 2 = Borrar
resultado	Entero	⮌	Código de error

Descripción

El comando *SMTP_To* añade la identidad de los destinatarios principales del mensaje. Todas las direcciones que figuran en el área de encabezado "Para" y "cc" de un mensaje electrónico son visibles para cada destinatario del mensaje.

smtp_ID es la referencia entero largo de un mensaje electrónico creado con el comando *SMTP_New*.

msgPara es un valor texto que contiene una o más direcciones electrónicas..

eliminarOpcion es un valor entero que especifica si desea añadir o eliminar el encabezado "Para":

- Un valor de cero, añade el nuevo valor al campo "Para".
- Si pasa un valor de 1, el contenido del parámetro pasado reemplaza el contenido del encabezado existente. En este caso, si pasa una cadena vacía en *msgPara*, el encabezado será removido).
- Si pasa un valor de 2, el encabezado "Para" se borra del mensaje.
  eliminarOpcion es un parámetro opcional que por defecto vale cero si no se especifica lo contrario.

Ejemplo

Consulte el ejemplo del comando *SMTP_Body*.

IC TCP/IP

- Rutinas de bajo nivel, Presentación
- TCP_Close
- TCP_Listen
- TCP_Open
- TCP_Receive
- TCP_ReceiveBLOB
- TCP_Send
- TCP_SendBLOB
- TCP_State

Rutinas de bajo nivel, Presentación

El protocolo TCP/IP (Transmission Control Protocol/Internet Protocol) es el protocolo principal utilizado para enviar y recibir datos a través de Internet. Los comandos Internet de bajo nivel de 4D permiten establecer sesiones TCP para enviar y recibir paquetes TCP a través de estas sesiones.

Hay dos maneras de establecer una conexión TCP. La primera consiste en ejecutar el comando *TCP_Open*. Esto abrirá una conexión con el dominio y puerto especificados. *TCP_Open* permite el uso del protocolo SSL (Secured Socket Layer), que permite una conexión segura. La segunda forma consiste en utilizar el comando *TCP_Listen*. Este comando abre una conexión con el dominio y el puerto especificados, y escucha las conexiones entrantes. La mejor manera de determinar si una conexión se ha establecido es comprobar el estado de la sesión con el comando *TCP_State* al finalizar el comando *TCP_Listen*. Un código de estado se devolverá, indicando el estado actual de la sesión. Desde aquí puede enviar y/o recibir paquetes TCP como lo haría con una conexión establecida con *TCP_Open*.

En todos los casos, toda conexión TCP abierta debe cerrarse posteriormente con ayuda del comando *TCP_Close*.

Los comandos TCP/IP de bajo nivel requieren conocimientos avanzados de los protocolos de comunicación. Los desarrolladores que utilizan estas rutinas deben tener una comprensión completa de todo protocolo que intenten implementar. Información sobre los diferentes números de puertos asignados a TCP/IP, protocolos de comunicación, requisitos de direccionamiento, etc. se pueden encontrar en las RFC.

Referencias de conexión en comandos TCP

4D Internet commands permite pasar directamente una referencia de conexión POP3, IMAP, o FTP a los comandos TCP de bajo nivel y viceversa.
De hecho, por un lado, los protocolos están en constante evolución lo que lleva a la creación de nuevos comandos; por otra parte, algunos paquetes de software hacen su propia interpretación de los RFCs, volviendo las implementaciones estándar inutilizables. Usando los comandos de bajo nivel TCP, los desarrolladores pueden crear las funciones de alto nivel que necesitan (en lugar de utilizar funciones existentes o para compensar una función que no existe).
Esto aumenta significativamente la compatibilidad y las posibilidades de desarrollo ya que los desarrolladores pueden crear sus propios comandos de alto nivel sin tener que reescribir todos los comandos necesarios para el uso de un protocolo.

En este ejemplo, el comando *IMAP_Capability* se reemplaza por una función equivalente desarrollada utilizando los comandos TCP_IP.

- Este es el método inicial utilizando el comando *IMAP_Capability*:

```
$ErrorNum:=IMAP_Login(vHost;vUserName;vUserPassword;vImap_ID)
If($ErrorNum=0)
   C_TEXT(vCapability)
   $ErrorNum:=IMAP_Capability(vImap_ID;vCapability)
   ... ` Comando IMAP utilizando el parámetro vImap_ID
End if
$ErrorNum:=IMAP_Logout(vImap_ID)
```

- Este método puede remplazarse por:

```
$ErrorNum:=IMAP_Login(vHost;vUserName;vUserPassword;vImap_ID)
If($ErrorNum =0)
   C_TEXT(vCapability)
 ` Método TCP utilizando el valor del parámetro vImap_ID:
   $ErrorNum:=My_IMAP_Capability(vImap_ID)
```

```
    ... ` Comandos IMAP utilizando el parámetro vImap_ID
End if
$ErrorNum:=IMAP_Logout(vImap_ID)
```

- Este es el código de la función ***My_IMAP_Capability***:

```
C_LONGINT($1;$vErrorNum;$0)
C_TEXT($vSentText;$vReceivedText;vCapability)
C_TEXT($2)

$vImap_Id:=$1
$vCmd_Id:="A001" ` Este ID de comando debe ser único (cf. RFC 2060)
$MyvtRequestCmd:="CAPABILITY"
$vSentText;:=$vCmd_Id+""+$MyvtRequestCmd+Character(13)+Character(10)
$vReceivedText:=""
$vErrorNum:=TCP_Send($vImap_Id;$vSentText)
If($vErrorNum=0)
   $vErrorNum:=TCP_Receive($vImap_Id;$vReceivedText)
   Case of
      :($vErrorNum#0) `Reception error
         vCapability:=""
      :(Position($vCmd_Id+" OK ";$vReceivedText)#0)
 ` Ejecución exitosa del comando
         vCapability:=$vReceivedText
 ` En este ejemplo, no procesamos la cadena recibida
      :(Position($vCmd_Id+" BAD ";$vReceivedText)#0)
 ` Falla de la ejecución del comando (error de sintaxis
 ` o comando desconocido)
         vCapability:=""
         $vErrorNum:=10096
   End case
End if
$0:=$vErrorNum
```

TCP_Close

TCP_Close (tcp_ID) -> resultado

Parámetro	Tipo		Descripción
tcp_ID	Entero largo	➡	Referencia de una sesión TCP abierta
		⬅	0 = Sesión cerrada con éxito
resultado	Entero	↩	Código de error

Descripción

El comando *TCP_Close* cierra la sesión TCP referenciada por *tcp_ID*.

Si una sesión TCP está abierta, ocupa una de las 64 referencias disponibles para las sesiones TCP. Si las 64 sesiones no se han cerrado, el usuario no podrá abrir otra sesión.

tcp_ID contiene la referencia de una sesión TCP abierta con el comando *TCP_Open* o *TCP_Listen*. Este comando devolverá el valor cero en el parámetro *tcp_ID* si la sesión se cierra correctamente.

TCP_Listen

TCP_Listen (direccionIP ; puertoRemoto ; puertoLocal ; timeout ; tcp_ID) -> resultado

Parámetro	Tipo		Descripción
direccionIP	Cadena	➡	Dirección IP local de escucha o "" para escuchar todas las direcciones disponibles
		⬅	Dirección IP distante utilizada (si se pasa una variable que contiene una cadena vacía)
puertoRemoto	Entero	➡	*** Parámetro ignorado ***
puertoLocal	Entero	➡	Número de puerto local, 0 = Utilizar un puerto local vacante
		⬅	Número de puerto local utilizado (si se pasa 0)
timeout	Entero	➡	Número de segundos de espera, 0 = No timeOut
tcp_ID	Entero largo	⬅	Referencia de la sesión TCP abierta
resultado	Entero	↩	Código de error

Descripción

El comando *TCP_Listen* abre un "socket" de comunicación en el puerto definido por los parámetros *direccionIP* y *puertoLocal*. Este comando no devuelve el control al método de llamada de 4D hasta que se realice una conexión o se haya superado el timeout. Aunque puede parecer que esto bloquea la base hasta que se logra la conexión, el comando es amigable con otros procesos 4D en ejecución. Este comando divide el tiempo con los otros procesos 4D en ejecución.

No se recomienda ejecutar *TCP_LISTEN* dentro de un proceso 4D particular (especialmente si no se especifica un *timeout*).

El parámetro *direccionIP* contiene la dirección IP utilizada para la conexión:

- Si pasa una cadena vacía en este parámetro, el comando escucha todas las direcciones disponibles en la máquina.
- Si pasa una variable que contiene una cadena vacía, el comando devolverá en el parámetro la dirección IP distante utilizada para la conexión.

puertoLocal contiene el número del puerto TCP a utilizar para la comunicación. Si pasa cero, el comando utilizará un puerto vacante y devolverá su número en este parámetro.

timeout especifica el número máximo de segundos que este comando esperará para una conexión entrante. Un cero en este parámetro hará que el comando espere indefinidamente por una conexión entrante. Utilice esta opción con precaución ya que el control no se devolverá al proceso de llamada 4D si no se realiza una conexión. Nunca pase cero en este parámetro si la base es de un solo proceso.

tcp_ID devuelve la referencia de la sesión abierta. Esta referencia será utilizada por todos los comandos TCP posteriores ejecutados en la sesión.

Cualquier conexión TCP abierta utilizando el comando *TCP_Listen* debe cerrarse más adelante utilizando el comando *TCP_Close*.

Ejemplo

```
C_LONGINT(vTCPID)
C_LONGINT(vStatus)
$err:=TCP_Listen("";0;0;30;vTCPID)
$err:=TCP_State(vTCPID;vStatus)
If(vStatus=2)  `El socket está abierto y escucha
   Hacercualquiercosa
   $err:=TCP_Close(vTCPID)
End if
```

TCP_Open

TCP_Open (nomServidor ; puertoRemoto ; tcp_ID ; paramSesion) -> resultado

Parámetro	Tipo		Descripción
nomServidor	Cadena	→	Nombre o dirección IP del servidor
puertoRemoto	Entero	→	Puerto remoto al cual conectarse (0=indiferente)
tcp_ID	Entero largo	←	Referencia de la sesión TCP abierta
paramSesion	Entero	→	Parámetros de la sesión TCP 0 = Síncrono (Valor por defecto) 1 = Asíncrono 2 = Modo SSL síncrono, 3 = Modo SSL Asíncrono
resultado	Entero	↩	Código de error

Descripción

El comando *TCP_Open* abre una conexión TCP de salida a un dominio.

TCP_Open establece una conexión con *nomServidor*, en el puerto referenciado por *puertoRemoto* (si este parámetro es diferente de 0). Un valor entero largo se devuelve en *tcp_ID*, el cual será utilizado por todas las llamadas TCP posteriores que hagan referencia a la sesión. Por defecto, una sesión abierta por *TCP_Open* tiene un timeout de 30 segundos si no se reciben datos por la sesión identificada por el parámetro *tcp_ID*. El valor del timeout por defecto puede cambiarse utilizando el comando *IT_SetTimeOut*.

nomServidor es el nombre del servidor o la dirección IP de la máquina con la cual usted abre una conexión.

puertoRemoto indica que el puerto TCP de la máquina indicada por nomServidor, con la cual desea establecer una conexión.

Nota: después de una llamada a *TCP_Open* (o *TCP_Listen*), *puertoRemoto* puede devolver un valor negativo si el valor pasado en este parámetro es superior a 32767. Esto no va a perturbar la conexión. Para solucionar este problema, puede utilizar una variable intermedia:

```
$v_ RemotePort:=v_ RemotePort
$err:=TCP_Open(v_ RemoteHostIPAdr;0;v_ SessionID)
```

tcp_ID devuelve la referencia de la sesión abierta. Esta referencia será utilizada por todos los comandos TCP posteriores ejecutados en la sesión.

El parámetro opcional *paramSesion* da al usuario la posibilidad de elegir la configuración de la sesión TCP. Tenga en cuenta que estos valores se aplican a cada comando TCP llamado durante la sesión. El valor por defecto es 0 (Síncrono, no SSL).

SSL (Secured Socket Layer) es un protocolo que permite establecer comunicaciones TCP seguras (consulte la documentación de 4D para obtener más información y requisitos de instalación).

Cualquier conexión TCP abierto utilizando el comando *TCP_Open* debe estar cerrada más adelante mediante el comando *TCP_Close*.

Asíncrono/Síncrono

En **modo asíncrono**, los comandos Internet de 4D devuelven el control al motor 4D inmediatamente después de su ejecución, sin esperar el final del proceso de conexión (sin esperar a que la conexión con el servidor remoto se establezca). El modo asíncrono es útil para las personas que no quieren que todos los comandos TCP utilicen tiempo de 4D.

En **modo síncrono**, los comandos Internet de 4D van al motor 4D (a otros procesos 4D) sólo cuando el proceso de conexión termina (con éxito o no).

- 0 = Modo Síncrono (modo por defecto, ejecuta como las versiones anteriores de 4D Internet Commands)
- 1 = Modo asíncrono
- 2 = SSL en uso, Síncrono. Todos los comandos TCP utilizando la referencia a esta sesión TCP (tcp_ID) se ejecutan en modo síncrono y utilizan el protocolo SSL.

- 3 = SSL en uso, Asíncrono. Todos los comandos TCP utilizando la referencia a esta sesión TCP (tcp_ID) se ejecutan en modo asíncrono y utilizan el protocolo SSL.

Nota: el error 10089 puede ser devuelto al pasar 2 o 3, si no se puede abrir una conexión SSL (librería SLI no encontrada en la carpeta 4D Extensions).

Ejemplo

Usted quiere conectarse a un sitio web utilizando Https; verifique que la librería SLI esté correctamente instalada y abra la conexión en el puerto 443:

```
$vError:=TCP_Open(hostName;443;tcp_ID;2)
...
$vError:=TCP_Close(tcp_ID)  `No se olvide de cerrar la sesión
```

TCP_Receive

TCP_Receive (tcp_ID ; texto) -> resultado

Parámetro	Tipo		Descripción
tcp_ID	Entero largo	➡	Referencia de una sesión TCP abierta
texto	Texto	⬅	Texto recibido
resultado	Entero	↩	Código de error

Descripción

El comando *TCP_Receive* permite recibir paquetes de datos a través de una sesión TCP.

tcp_ID es una referencia entero largo a una sesión TCP abierta con el comando *TCP_Open* o *TCP_Listen*.

texto es el texto recibido. Al recibir datos a través de paquetes TCP, no puede contar con que todos sus datos sean recibidos por una sola llamada *TCP_Receive*. El comando *TCP_Receive* se suele llamar dentro de un bucle Repeat que continuamente verifica el estado de la conexión o está buscando un valor particular.

Ejemplo

```
C_LONGINT($tcp_id)
C_TEXT($webpage;$buffer)
C_LONGINT(vState;$error)
$webpage:=""
vState:=0
Repeat
   $error:=TCP_Receive($tcp_id;$buffer)
   $error:=TCP_State($tcp_id;vState)
   $webpage:=$webpage+$buffer
Until((vState=0)|($error#0))hasta que el servidor cierre la conexión o se presente un error
```

TCP_ReceiveBLOB

TCP_ReceiveBLOB (tcp_ID ; datosRecibidos) -> resultado

Parámetro	Tipo		Descripción
tcp_ID	Entero largo	➡	Referencia de una sesión TCP abierta
datosRecibidos	BLOB	⬅	BLOB para recibir los datos
resultado	Entero	↩	Código de error

Descripción

El comando *TCP_ReceiveBLOB* recibe paquetes de datos a través de una sesión TCP.

Este comando funciona igual que el comando *TCP_Receive*, con la diferencia de que recibe datos en un BLOB en lugar de un texto, permitiendo superar el límite de 32K de los datos de tipo texto y recibir objetos binarios.

tcp_ID es una referencia entero largo a una sesión TCP abierta con el comando *TCP_Open* o *TCP_Listen*.

blobARecibir es el BLOB que recibe los datos. Al recibir datos a través de paquetes TCP, no puede contar con que todos sus datos sean recibidos por una sola llamada *TCP_ReceiveBLOB*. El comando *TCP_ReceiveBLOB* se suele llamar dentro de un bucle **Repeat...Until** que verifica continuamente el estado de la conexión o busca un valor particular.

Ejemplo

Este ejemplo muestra la estructura típica de un método que utiliza el comando *TCP_ReceiveBLOB*:

```
C_BLOB($Blob_Received;$Blob_All)
C_LONGINT($srcpos;$dstpos)
Repeat
   $Err:=TCP_ReceiveBLOB($TCP_ID;$Blob_Received )
   $Err:=TCP_State($TCP_ID;$State)
   $srcpos:=0
   $dstpos:=BLOB size($Blob_All)
  `Concatenating received Blobs
   COPY BLOB($Blob_Received;$Blob_All;$srcpos;$dstpos;BLOB size($Blob_Received))
Until(($State=0)|($Err#0))
```

TCP_Send

TCP_Send (tcp_ID ; textoAEnviar) -> resultado

Parámetro	Tipo		Descripción
tcp_ID	Entero largo	➡	Referencia de una sesión TCP abierta
textoAEnviar	Texto	➡	Texto a enviar
resultado	Entero	↩	Código de error

Descripción

El comando *TCP_Send* envía los datos a la sesión TCP designada por *tcp_ID*.

tcp_ID es la referencia de una sesión TCP abierta, establecida con el comando *TCP_Open* o *TCP_Listen*.

El parámetro *enviarTexto* contiene un valor de tipo texto a enviar a la sesión TCP referenciada por *tcp_ID*.

Nota: al enviar texto no us-ascii o texto largo, se recomienda utilizar **TCP_SendBLOB** porque nos permite un mejor control de la codificación del texto.

TCP_SendBLOB

TCP_SendBLOB (tcp_ID ; blobAEnviar) -> resultado

Parámetro	Tipo		Descripción
tcp_ID	Entero largo	➡	Referencia de una sesión TCP abierta
blobAEnviar	BLOB	➡	Blob a enviar
resultado	Entero	⮌	Código de error

Descripción

El comando *TCP_SendBLOB* envía los datos a la sesión TCP designada por tcp_ID. este comando funciona igual que el comando *TCP_Send*, excepto que envía un BLOB en lugar de un texto, lo cual permite superar la limitación de 32K de los datos de tipo texto y enviar objetos binarios.

tcp_ID es una referencia entero largo a una sesión TCP abierta con el comando *TCP_Open* o *TCP_Listen*.

blobAEnviar es el BLOB a enviar a la sesión TCP referenciada por *tcp_ID*.

Nota sobre la independencia de plataforma: si intercambia BLOBs entre plataformas Macintosh y PC, depende de usted tratar las conversiones de bytes ("byte swapping"), si es necesario.

Ejemplo

Este ejemplo envía un BLOB en la sesión TCP:

```
C_BLOB($Blob_Send)
C_TEXT(v_Txt_Send)
TEXT TO BLOB(v_Txt_Send;$Blob_Send;Text without length;*)
$err:=TCP_SendBLOB(v_tcp_ID;$Blob_Send)
```

TCP_State

TCP_State (tcp_ID ; codigoEstado) -> resultado

Parámetro	Tipo		Descripción
tcp_ID	Entero largo	➡	Referencia de una sesión TCP abierta
codigoEstado	Entero	⬅	Código del estado TCP
resultado	Entero	↩	Código de error

Descripción

El comando *TCP_State* devuelve un valor entero que indica el estado de una conexión TCP particular. *tcp_ID* contiene la referencia de una sesión TCP abierta con el comando *TCP_Open* o *TCP_Listen*.
El parámetro *codigoEstado* es una variable que devuelve uno de los siguientes códigos de estado.

0 Conexión cerrada
2 Escucha de una conexión entrante
8 Conexión establecida

Ejemplo

Este ejemplo asume que se estableció una conexión TCP válida y se identifica por la variable $tcp_id. En este ejemplo, se envía un comando a un servidor web para solicitar una página de información y luego se introduce un bucle para recibir los resultados. Como los servidores web cierran automáticamente las conexiones una vez realizan su acción, este ejemplo continúa recibiendo los datos hasta que la conexión se detiene o se produce un error.

```
C_LONGINT($tcp_id)
C_LONGINT(vState;$err)
C_TEXT($command;$buffer;$response)
If(TCP_Send($tcp_id;$command)=0)
   vState:=0
   Repeat
      $err:=TCP_Receive($tcp_id;$buffer)
      $err:=TCP_State($tcp_id;vState)
      $response:=$response+$buffer
   Until((vState=0)|($err#0))
End if
```

IC UDP

- Comandos UDP, Presentación
- UDP_Delete
- UDP_New
- UDP_ReceiveBLOBFrom
- UDP_ReceiveFrom
- UDP_SendBLOBTo
- UDP_SendTo

Comandos UDP, Presentación

UDP (User Datagram Protocol) es un protocolo de comunicaciones que permite la transmisión de datos por paquetes. Es más fácil de implementar y más rápido que el protocolo TCP (el encabezado UDP tiene 8 bytes, en comparación con mínimo 20 bytes del encabezado TCP), pero no el mismo nivel de confiabilidad. Es útil para aplicaciones donde la velocidad de transmisión de datos es importante. Sin embargo, no verifica que los paquetes lleguen a su destino y no ofrece el sistema de control de errores ni del sistema de recuperación.

Ejemplo

El siguiente ejemplo utiliza el protocolo UDP para recuperar en los arrays la lista de servidores 4D activos en la red local:

```
ARRAY TEXT(255;asHost;0)
ARRAY TEXT(32;asMachineName;0)
ARRAY TEXT(32;asService;0)
ARRAY TEXT(32;asDBName;0)
C_BLOB($Blob)

$Addr:="255.255.255.255"
$Port:=19813
$Offset:=32
SET BLOB SIZE($Blob;96;0)
TEXT TO BLOB("4D Server II";$Blob;Mac text without length;$Offset)

$Err:=UDP_New(0;$udpID)
$Err:=UDP_SendBLOBTo($udpID;$Addr;$Port;$Blob)
$Secs:=5
$Timeout:=Milliseconds+($Secs*1000)
Repeat
   DELAY PROCESS(Current process;6)  `... en tics
   SET BLOB SIZE($Blob;0;0)
   $PeerAddr:=$Addr
   $Err:=UDP_ReceiveBLOBFrom($udpID;$PeerAddr;$Port;$Blob)

   If(BLOB size($Blob)>0)
      $Offset:=0
      $Host:=BLOB to text($Blob;Mac C string;$Offset;32)
      $Offset:=32
      $Service:=BLOB to text($Blob;Mac C string;$Offset;32)
      $Offset:=64
      $DBName:=BLOB to text($Blob, C string;$Offset;32)
      $Pos:=Find in array(asMachineName;$Host)
      If($Pos=-1)
         APPEND TO ARRAY(asHost;$PeerAddr)
         APPEND TO ARRAY(asMachineName;$Host)
         APPEND TO ARRAY(asService;$Service)
         APPEND TO ARRAY(asDBName;$DBName)
      End if
   End if
```

```
Until((Milliseconds>$Timeout)|($Err#0))
$Err:=UDP_Delete($udpID)
```

UDP_Delete

UDP_Delete (udp_ID) -> resultado

Parámetro	Tipo		Descripción
udp_ID	Entero largo	➡	Referencia de conexión UDP
resultado	Entero	↩	Código de error

Descripción

El comando *UDP_Delete* cierra la conexión UDP referenciada por *udp_ID*, previamente creada con ayuda del comando *UDP_New*.

UDP_New

UDP_New (puertoLocal ; udp_ID) -> resultado

Parámetro	Tipo		Descripción
puertoLocal	Entero	➡	Puerto local, 0 = Utilizar un puerto local vacante
udp_ID	Entero largo	⬅	Referencia de conexión UDP
resultado	Entero	↩	Código de error

Descripción

El comando *UDP_New* permite crear una conexión (socket) UDP en el puerto pasado en el parámetro *puertoLocal*.

Si pasa 0, el comando buscará un puerto sin utilizar. La referencia de la conexión UDP abierta se devuelve en el parámetro *udp_ID*.

Cuando esta conexión no sea necesaria, recuerde cerrarla utilizando el comando *UDP_Delete* para liberar memoria.

UDP_ReceiveBLOBFrom

UDP_ReceiveBLOBFrom (udp_ID ; nomServidor ; puertoRemoto ; BLOB) -> resultado

Parámetro	Tipo		Descripción
udp_ID	Entero largo	➡	Referencia de conexión UDP
nomServidor	Variable texto	⬅	Nombre o dirección IP del servidor que responde
puertoRemoto	Variable entero largo	⬅	Puerto del servidor remoto que responde
BLOB	BLOB	⬅	BLOB recibido
resultado	Entero	↩	Código de error

Descripción

El comando **UDP_ReceiveBLOBFrom** permite recibir un BLOB enviado por la conexión *udp_ID*.

Pase variables en los parámetros *nomServidor* y *remotePort*. Después de ejecutar el comando, estas variables contendrán, respectivamente, el nombre (o dirección IP) y el número de puerto del servidor remoto del cual se recibió el BLOB.

El BLOB recibido se devuelve en el parámetro *blob*.

UDP_ReceiveFrom

UDP_ReceiveFrom (udp_ID ; nomServidor ; puertoRemoto ; texto) -> resultado

Parámetro	Tipo		Descripción
udp_ID	Entero largo	➡	Referencia de conexión UDP
nomServidor	Variable texto	⬅	Name or IP address of server that responds
puertoRemoto	Variable entero largo	⬅	Port of remote server that responds
texto	Texto	⬅	Texto recibido
resultado	Entero	↩	Código de error

Descripción

El comando **UDP_ReceiveFrom** permite recibir el texto enviado por la conexión *udp_ID*.

Pase las variables en los parámetros *nomServidor* y *puertoRemoto*. Después de ejecutar el comando, estas variables contendrán, respectivamente, el nombre (o dirección IP) y el número de puerto del servidor remoto desde el que se recibió el texto.

El texto recibido se devuelve en el parámetro *texto*.

UDP_SendBLOBTo

UDP_SendBLOBTo (udp_ID ; nomServidor ; puertoRemoto ; blobAEnviar) -> resultado

Parámetro	Tipo		Descripción
udp_ID	Entero largo	→	Referencia de conexión UDP
nomServidor	Cadena	→	Nombre o dirección IP del servidor
puertoRemoto	Entero	→	Puerto remoto a utilizar (0=indiferente)
blobAEnviar	BLOB	→	BLOB a enviar
resultado	Entero	↩	Código de error

Descripción

El comando *UDP_SendBLOBTo* permite enviar un BLOB utilizando la conexión UDP abierta referenciada en el parámetro *udp_ID*.

nomServidor el nombre o la dirección IP del servidor al cual los datos deben enviarse.

puertoRemoto es el número del puerto a utilizar. Si pasa 0, se utilizará cualquier puerto disponible.

blobAEnviar contiene el BLOB a enviar.

UDP_SendTo

UDP_SendTo (udp_ID ; nomServidor ; puertoRemoto ; textoAEnviar) -> resultado

Parámetro	Tipo		Descripción
udp_ID	Entero largo	➡	Referencia de conexión UDP
nomServidor	Cadena	➡	Nombre o dirección IP del servidor
puertoRemoto	Entero	➡	Puerto remoto a utilizar (0=indiferente)
textoAEnviar	Texto	➡	Texto a enviar
resultado	Entero	↩	Código de error

Descripción

El comando *UDP_SendTo* permite enviar datos texto vía la conexión UDP abierta referenciada en el parámetro *udp_ID*.

nomServidor el nombre o la dirección IP del servidor al cual los datos deben enviarse.

puertoRemoto es el número del puerto a utilizar. Si pasa 0, se utilizará cualquier puerto disponible.

textoAEnviar es el texto a enviar.

IC Utilities

Utilitarios, Presentación
IT_Decode
IT_Encode
IT_ErrorText
IT_GetPort
IT_GetProxy
IT_GetTimeOut
IT_MyTCPAddr
IT_Platform
IT_PPPConnect
IT_PPPDisconnect
IT_PPPStatus
IT_SetPort
IT_SetProxy
IT_SetTimeOut
IT_TCPversion
IT_Version
IT_MacTCPInit
IT_MacTCPVer

Utilitarios, Presentación

Los comandos de esta sección ofrecen diversas utilidades que soportan otras secciones de 4D Internet Commands. Muchos de estos comandos ayudan a los desarrolladores a determinar el entorno en el que la máquina del usuario está funcionando, las versiones del software utilizado o el estado actual y la dirección IP de su ordenador.
Otros comandos dentro de esta sección ayudan al desarrollador a descifrar códigos de error, codificar y decodificar archivos, y a actuar en el valor máximo de espera (timeout) por defecto aplicable a la mayoría de los comandos de otros temas.

IT_Decode

IT_Decode (nomArchivo ; nomArchivoDecod ; modoDecod) -> resultado

Parámetro	Tipo		Descripción
nomArchivo	Texto	➡	Ruta de acceso local a un archivo codificado
nomArchivoDecod	Texto	➡	Ruta de acceso del archivo decodificado
		⬅	Ruta de acceso resultante del archivo decodificado
modoDecod	Entero	➡	1 = BinHex 2 = Base64 (Data fork únicamente) 3 = AppleSingle 4 = AppleDouble 5 = AppleSingle Y Base64 6 = AppleDouble Y Base64 7 = UUEncode 8 = MacBinary
resultado	Entero	↩	Código de error

Descripción

El comando *IT_Decode* decodifica un archivo utilizando el *modoDecod* especificado. El archivo especificado no se modifica y se crea una copia decodificada.

nomArchivo contiene la ruta de acceso completa del archivo a decodificar.

El parámetro *nomArchivoDecod* puede contener:

- La ruta de acceso completa de la copia decodificada del archivo, lo que permite definir el nombre y la ubicación.
- La ruta completa de la carpeta que va a recibir el archivo decodificado utilizando el nombre del archivo original.
- Una cadena vacía (en este caso, el comando *IT_Decode*) asigna un nombre para el documento, que se ubica en la misma carpeta que el archivo especificado en el primer parámetro.
  Si se especifica o no, la ruta completa del documento decodificado se devolverá en este parámetro.

modoDecod indica el método de decodificación a aplicar al archivo. El valor por defecto 1(decodificación binhex). Otros métodos son los siguientes:

Código	Método	Disponible en
1	BinHex	Win y Mac
2	Base64 (Data fork only)	Win y Mac
3	AppleSingle	Mac
4	AppleDouble	Mac
5	AppleSingle and Base64	Mac
6	AppleDouble and Base64	Mac
7	UUEncode	Win y Mac
8	MacBinary	Win y Mac

Para la decodificación utilizando AppleDouble (modos de decodificación 4 y 6), este comando busca un archivo llamado "%nomArchivo" para el resource fork.

IT_Encode

IT_Encode (nomArchivo ; nomArchivoCod ; modoCod) -> resultado

Parámetro	Tipo		Descripción
nomArchivo	Texto	→	Ruta de acceso local a un archivo
nomArchivoCod	Texto	→	Ruta de acceso del archivo codificado
		←	Ruta de acceso al archivo codificado resultante
modoCod	Entero	→	Mac y Win: 1 = BinHex 2 = Base64 (Data fork únicamente) 7 = UUEncode 8 = MacBinary; Mac únicamente: 3 = AppleSingle 4 = AppleDouble 5 = AppleSingle Y Base64 6 = AppleDouble Y Base64
resultado	Entero	↩	Código de error

Descripción

El comando *IT_Encode* codifica un archivo utilizando el *modoCod* especificado. El archivo especificado no se modificará y se crea una copia codificada. El nombre del archivo codificado creado será el nombre del archivo original más un sufijo para especificar el método de codificación. Para la codificación BinHex, se añadirá el sufijo ".hqx". Para la codificación Base64, se añadirá el sufijo ".b64". Para la codificación AppleSingle, se añadirá el sufijo ".as".

nomArchivo contiene la ruta de acceso completa del archivo a codificar.

nomArchivoCod puede contener:

- La ruta de acceso completa de la copia codificada del archivo, lo que permite definir el nombre y la ubicación.
- La ruta completa de acceso de la carpeta que va a recibir el archivo codificado (sin especificar el nombre del archivo); el nombre del archivo será el nombre del archivo original con un sufijo para especificar el método de codificación.
- Una cadena vacía (en este caso, el comando *IT_Encode*) asigna un nombre para el documento, que se ubica en la misma carpeta que el archivo especificado en el primer parámetro. Si se especifica o no, la ruta completa del documento codificado se devolverá en este parámetro. Debido a la posibilidad de posibles conflictos de nombres en el directorio especificado, se recomienda basarse en el valor resultante del parámetro.

modoCod define el método de codificación a aplicar al archivo. El valor por defecto es 1 para la codificación BinHex. Otros métodos son los siguientes:

Código	Método	Disponible en
1	BinHex	Win y Mac
2	Base64 (Data fork only)	Win y Mac
3	AppleSingle	Mac
4	AppleDouble	Mac
5	AppleSingle and Base64	Mac
6	AppleDouble and Base64	Mac
7	UUEncode	Win y Mac
8	MacBinary	Win y Mac

Para la codificación utilizando AppleDouble (modos de codificación 4 y 6), se crean dos archivos llamados "%nomArchivo" y "nomArchivo".

IT_ErrorText

IT_ErrorText (error) -> Resultado

Parámetro	Tipo		Descripción
error	Entero	→	Código del error
Resultado	Cadena	↩	Texto del error

Descripción

El comando *IT_ErrorText* devuelve la descripción del error cuyo número se pasa como parámetro. Note que este es uno de los pocos 4D Internet Commands que no devuelven un entero como valor.

error contiene el número del error.

Ejemplo

El siguiente es un ejemplo de una rutina ***ErrorCheck*** que muestra un mensaje de alerta explicando la causa del error.

```
  `Method: ERRCHECK ("Command Name"; Error# ) -> True/False
C_TEXT(vErrorMsg)
$Command:=$1
$Error:=$2
$Result:=True
If($Error#0)
   $Result:=False
   vErrorMsg:=IT_ErrorText($Error)
   ALERT("ERROR -- "+Char(13)+"Command: "+$Command+Char(13)+"Error Code:"+String($Error)
   +Char(13)+"Description: "+vErrorMsg)
End if
$0:=$Result
```

IT_GetPort

IT_GetPort (protocolo ; puerto) -> resultado

Parámetro	Tipo		Descripción
protocolo	Entero	➡	1 = FTP; 2 = SMTP; 3 = POP3; 4 = IMAP; 12 = SMTP SSL ; 13 = POP3 SSL ; 14 = IMAP SSL
puerto	Entero	⬅	Número de puerto
resultado	Entero	↩	Código de error

Descripción

Para el *protocolo* especificado, el comando *IT_GetPort* recupera el número del *puerto* actual utilizado por los 4D Internet Commands relativos al protocolo.

IT_GetProxy

IT_GetProxy (protocolo ; tipoProxy ; nomServidorProxy ; puertoProxy ; idUsuarioProxy) -> resultado

Parámetro	Tipo		Descripción
protocolo	Entero	➡	1 = FTP; 2 = SMTP; 3 = POP3; 4 = IMAP
tipoProxy	Entero	⬅	0 = Ninguno; 1 = SOCKS
nomServidorProxy	Cadena	⬅	Nombre o dirección IP del servior proxy SOCKS
puertoProxy	Entero	⬅	Puerto del proxy al cual conectarse
idUsuarioProxy	Texto	⬅	ID del usuario para SOCKS
resultado	Entero	↩	Código de error

Descripción

El comando *IT_GetProxy* devuelve los parámetros actuales utilizados por los 4D Internet Commands relacionados con el enrutamiento del protocolo especificado. Los valores estarán en su estado por defecto a menos que una llamada anterior a *IT_SetProxy* alteré la configuración. Para obtener una descripción completa de los parámetros, consulte *IT_SetProxy*.

protocolo es un valor entero que especifica el protocolo a examinar. Un valor de 1 indica el protocolo FTP. Un valor de 2 indica el protocolo SMTP. Un valor de 3 indica el protocolo POP3. Un valor de 4 indica el protocolo IMAP.

El parámetro *tipoProxy* devuelve la configuración actual para determinar se utiliza un servidor proxy SOCKS Un valor de 1 enruta todas las peticiones para el protocolo especificado a través del servidor SOCKS especificado. Un valor de cero no enruta las peticiones para el protocolo especificado a través de un servidor SOCKS.

nomServidorProxy devuelve el nombre o la dirección IP del servidor proxy SOCKS en uso.

El parámetro *puertoProxy* devuelve el número de puerto utilizado con el protocolo especificado para comunicarse con el servidor proxy SOCKS.

El parámetro *IDUsuarioProxy* devuelve el ID del usuario.

IT_GetTimeOut

IT_GetTimeOut (timeout) -> resultado

Parámetro	Tipo		Descripción
timeout	Entero	⇐	Tiempo de espera en segundos
resultado	Entero	⇒	Código de error

Descripción

El comando *IT_GetTimeOut* devuelve el valor actual del timeout para los comandos listados en *IT_SetTimeOut*.

timeout devuelve el valor actual en segundos del tiempo de espera.

IT_MyTCPAddr

IT_MyTCPAddr (direccion_IP ; subred) -> resultado

Parámetro	Tipo		Descripción
direccion_IP	Cadena	⬅	Dirección IP de la máquina del usuario
subred	Cadena	⬅	Máscara de subred en formato IP
resultado	Entero	↩	Código de error

Descripción

El comando *IT_MyTCPAddr* devuelve la dirección IP de la máquina que ejecuta el comando.

El parámetro *direccion_IP* devuelve una cadena de caracteres que contiene la dirección IP.

El parámetro *sured* devuelve una cadena de caracteres que contiene la máscara de subred de la dirección IP.

IT_Platform

IT_Platform -> resultado

Parámetro	Tipo		Descripción
resultado	Entero		Tipo de plataforma (1 = Mac OS, 2 = Windows)

Descripción

La función *IT_Platform* devuelve un entero indicando la plataforma en uso.
La función devuelve 1 para Mac OS y 2 para Windows.

Ejemplo

```
C_BOOLEAN(◇ITnative)
◇ITnative:=(IT_Platform=1)
```

IT_PPPConnect

IT_PPPConnect (profilPPP) -> resultado

Parámetro	Tipo		Descripción
profilPPP	Cadena	➡	Nombre de conexión a distancia (cadena vacía bajo Mac OS, nombre bajo Windows
resultado	Entero	⮌	Código de error

Descripción

El comando *IT_PPPConnect* abre la conexión a distancia actual bajo Mac OS o la conexión a distancia especificada vía el parámetro (*pppProfil*) bajo Windows. Este comando actúa como una función y devuelve un código de error si la conexión no se puede abrir.

Este comando debe ejecutarse cada vez que necesite ejecutar un conjunto de comandos 4DIC que trabajan en línea. Al terminar la sesión, debe ejecutar *IT_PPPDisconnect* para cerrar la conexión actual.

El protocolo PPP (Point-to-Point Protocol) se utiliza para la comunicación entre dos ordenadores utilizando una interfaz serial, normalmente un ordenador personal conectado por una línea telefónica a un servidor. Por ejemplo, su proveedor de servidor de Internet puede ofrecerle una conexión PPP de manera que el servidor del proveedor pueda responder a sus peticiones, pasarlas a Internet y reenviar sus respuestas de nuevo a usted. Esquemáticamente, el protocolo encapsula los paquetes TCP/IP enviados por su máquina y los reenvía al servidor donde pueden servirse en la Internet.

PPP es generalmente preferible al antiguo estándar "de facto" SLIP (Serial Line Internet Protocol), ya que puede manejar la comunicación síncrona y asíncrona. PPP puede compartir una línea con otros usuarios y tiene una función de detección de errores, ninguno de los cuales es cierto para el protocolo SLIP. Si puede elegir, prefiera PPP.

IT_PPPDisconnect

IT_PPPDisconnect (profilPPP) -> resultado

Parámetro	Tipo		Descripción
profilPPP	Cadena	➡	Nombre de conexión a distancia (cadena vacía bajo Mac OS, nombre bajo Windows)
resultado	Entero	⮌	Código de error

Descripción

El comando *IT_PPPDisconnect* cierra la conexión a distancia actual abierta previamente por *IT_PPPConnect*.

profilPPP es un valor texto que especifica la conexión a cerrar.
Bajo Windows, este parámetro es útil cuando varias conexiones PPP se abren al mismo tiempo. El uso de este parámetro garantiza la ejecución del comando independientemente de la configuración de red del usuario.

Bajo Windows

- Si una sola conexión PPP está abierta y si el parámetro *profilPPP* se omite o contiene una cadena vacía, *IT_PPPDisconnect* cierra la conexión abierta.
- Si varias conexiones PPP están abiertas y si el parámetro *profilPPP* no se pasa o si contiene una cadena vacía, *IT_PPPDisconnect* devuelve un error y no cierra ninguna conexión.
- Si se pasa *profilPPP* y es válido, la conexión específica se cierra, sin importar el número de conexiones abiertas.

Bajo Mac OS

Este parámetro no se tiene en cuenta.

IT_PPPStatus

IT_PPPStatus (profilPPP) -> resultado

Parámetro	Tipo		Descripción
profilPPP	Cadena	➡	Nombre de conexión a distancia (cadena vacía bajo Mac OS, nombre bajo Windows
resultado	Entero	➲	1 = conectado; 0 = en conexión; -1 = error

Descripción

El comando *IT_PPPStatus* le permite probar el estado de una conexión abierta por el comando *IT_PPPConnect* o manualmente.

El parámetro *profilPPP* especifica la conexión a probar.
Bajo Windows, este parámetro es opcional pero puede ser útil para garantizar la ejecución del comando independientemente de la configuración de red del usuario.

Bajo Windows

- Si se pasa *profilPPP* y es válido, se devuelve el estado de la conexión especificada.
- Si *profilPPP* no se pasa o contiene una cadena vacía, *IT_PPPStatus* devuelve:
  - -1 si varias conexiones están abiertas,
  - el estado de la conexión abierta si sólo una conexión está abierta

Bajo Mac OS

Este parámetro no se tiene en cuenta.
IT_PPPStatus devuelve un entero indicando el estado de la conexión:

- 1 si está conectado,
- 0 si se está conectando,
- -1 en caso de una falla en la conexión o si no está conectado.

Ejemplo

```
  `Método GetMessages (método ejecutado en un proceso)
If(mPPPConnect($vPPPProfil;120))
   $vErrCode:=IT_MacTCPInit
   If($vErrCode=0)
      $vErrCode:=POP3_Login...
      ...
   Else
      ALERT("Connection failed")
   End if
End if

  `Método mPPPConnect
C_BOOLEAN($0)  `devuelve True si estamos conectados actualmente, False si la conexión falla
C_TEXT($1)  `cadena vacía bajo Mac OS, Nombre bajo Windows
C_LONGINT($2)  `timeout en segundos

If(IT_PPPStatus=1)
   $0:=True  `ya estamos conectados
Else
```

```
    $vTimeoutLength:=$2
    $vTimeout:=False
    $vErr:=IT_PPPConnect($1)
    If($vErr=0)
        $vStart:=Current time
        Repeat
            DELAY PROCESS(Current process;30)
            $vStatus:=IT_PPPStatus($1)
            $vTimeout:=((Current time-$vStart)>$vTimeoutLength)
        Until(($vStatus=1)|$vTimeout) `Conexión o time out
        If(Not($vTimeout))
            $0:=True `Conexión
        End if
    End if `... $Err = 0
End if
```

IT_SetPort

IT_SetPort (protocolo ; puerto) -> resultado

Parámetro	Tipo		Descripción
protocolo	Entero	→	1 = FTP ; 2 = SMTP ; 3 = POP3 ; 4 = IMAP ; 12 = SMTP TLS ; 13 = POP3 TLS ; 14 = IMAP TLS
puerto	Entero	→	Número de puerto
resultado	Entero	↩	Código de error

Descripción

Para el *protocol*o especificado, el comando *IT_SetPort* dirige todas las comunicaciones posteriores del protocolo al *puerto* especificado.

IT_SetProxy

IT_SetProxy (protocolo ; tipoProxy ; nomServidorProxy ; puertoProxy ; idUsuarioProxy) -> resultado

Parámetro	Tipo		Descripción
protocolo	Entero	➡	1 = FTP; 2 = SMTP; 3 = POP3; 4 = IMAP
tipoProxy	Entero	➡	0 = Ninguno; 1 = SOCKS
nomServidorProxy	Cadena	➡	Nombre o dirección IP del servidor proxy SOCKS
puertoProxy	Entero	➡	Puerto del proxy al cual conectarse
idUsuarioProxy	Texto	➡	ID de usuario para SOCKS
resultado	Entero	↩	Código de error

Descripción

El comando *IT_SetProxy* permite abrir una conexión utilizando el protocolo especificado para luego enviar todas las peticiones a través del servidor SOCKs (Proxy SOCKS). Si se acaba de conectar a una intranet, entonces probablemente no necesite comunicarse a través del servidor SOCKS. Sin embargo, todo depende de cómo su empresa tiene configurado el firewall. La configuración del *IT_SetProxy* tiene un alcance interprocesos y afecta todas las conexiones utilizando el protocolo especificado en cualquier proceso 4D.

Nota: Socks (o "SOCKS") es un protocolo que un servidor proxy puede utilizar para aceptar las peticiones de los usuarios cliente de una empresa para que pueda transmitirlas en Internet. Si su estación de trabajo se encuentra detrás de un firewall y desea acceder a una información que se encuentra en el Internet, el servidor SOCKES recibe su petición, la transmite a través del firewall y luego devuelve la información a la aplicación cliente.

protocolo es un valor entero que especifica el protocolo que debe ser filtrado por el servidor proxy SOCKS especificado. Pase 1 para designar el protocolo FTP. 2 para el protocolo SMTP. 3 para el protocolo POP3. 4 para el protocolo IMAP.

tipoProxy es un valor entero que indica si el protocolo especificado debe ser enrutado a través de un servidor proxy SOCKS o no. Pase 1 para enrutar todas las peticiones del protocolo por el servidor SOCKS especificado, de lo contrario pase 0.

nomServidorProxy es el nombre del servidor o la dirección IP del servidor proxy SOCKS.

puertoProxy es un valor entero que especifica el puerto a usar para que el protocolo especificado se comunique con el servidor proxy SOCKS.

IDUsuarioProxy es un valor texto que identifica al usuario. El ID de usuario es dado por su administrador de red. *IDUsuarioProxy* puede ser un texto vacío ("").

Ejemplo

El siguiente método permite enrutar todas las conexiones FTP vía el servidor Proxy SOCKS especificado.

```
$err:=IT_SetProxy(1;1;$proxyAdd;$proxyPort;"") `FTP SOCKS Proxy
$err:=FTP_Login("ftp.4d.com";"anonymous";dbody@aol.com";$ftpID)
$err:=FTP_GetFileInfo($ftpID;$vpath;$vsize;$vmodDate)
$err:=FTP_Receive($ftpID;$vpath;"";0)
$err:=FTP_Logout($ftpID)
```

Nota: por claridad, este ejemplo no contiene verificación de errores.

La siguiente instrucción detiene el enrutamiento de las conexiones FTP por el servidor proxy SOCKS.

```
$err:=IT_SetProxy(1;0;$proxyAdd;$proxyPort;"")
```

IT_SetTimeOut

IT_SetTimeOut (timeout) -> resultado

Parámetro	Tipo		Descripción
timeout	Entero	→	Timeout en segundos; de 0 a 127
resultado	Entero	↩	Código de error

Descripción

El comando *IT_SetTimeOut* permite definir el valor del timeout en segundos. Este valor está entre cero y 127 segundos. Por defecto, el periodo de timeout es de 30 segundos.

timeout es el valor actual en segundos del periodo timeout. Los siguientes comandos están afectados por *IT_SetTimeOut*:

TCP_Open
FTP_Login
FTP_Send
FTP_Receive
SMTP_QuickSend
SMTP_Send
POP3_Login
POP3_BoxInfo
POP3_Delete
POP3_Reset
POP3_MsgInfo
POP3_MsgLstInfo
POP3_GetMessage
POP3_MsgLst
POP3_Download
POP3_VerifyID
POP3_UIDToNum
IMAP_Login
IMAP_VerifyID
IMAP_Capability
IMAP_ListMBs
IMAP_SubscribeMB
IMAP_GetMBStatus
IMAP_SetCurrentMB
IMAP_Delete
IMAP_MsgInfo
IMAP_MsgLstInfo
IMAP_GetMessage
IMAP_MsgLst
IMAP_SetFlags
IMAP_GetFlags
IMAP_Search
IMAP_MsgFetch
IMAP_Download
IMAP_CopyToMB
IMAP_CreateMB
IMAP_RenameMB
IMAP_DeleteMB
NET_Finger
NET_Ping
NET_Time

Nota: la definición del *timeout* en cero para el comando *TCP_Listen* permite escucharlo indefinidamente. Asegúrese de pasar a otro valor después de utilizar este comando. Igualmente, el valor del timeout se utiliza para los "timeouts TCP/IP" Y "timeout de espera de respuesta". Si define el timeout en cero, no tendrá tiempo para esperar la respuesta.

IT_TCPversion

IT_TCPversion (tipoPila ; versionPila) -> resultado

Parámetro	Tipo		Descripción
tipoPila	Entero	⇐	0 = Ninguno, 3 = WinSock, 4 = BSD
versionPila	Texto	⇐	Número de versión de la pila TCP
resultado	Entero	⇒	Código de error

Descripción

El comando *IT_TCPversion* devuelve la información sobre el tipo de pila TCP en uso por los 4D Internet Commands. El tipo de pila varía según la plataforma. Bajo Macintosh, sólo se soporta la pila TCP BSD. En Windows, se soporta la pila TCP WinSock.

El parámetro *tipoPila* devuelve un valor entero que expresa el tipo de pila TCP en uso. El valor devuelto identifica las siguientes pilas TCP soportadas:

Código	Pila TCP
0	Ninguno
1	MacTCP (Obsoleto, ver las notas de compatibilidad)
2	Open Transport (Obsoleto, ver las notas de compatibilidad)
3	WinSock
4	BSD Sockets

Notas de compatibilidad:

- A partir de la versión 6.8 de 4D Internet Commands, MacTCP ya no se soporta. Por lo tanto, el parámetro *tipoPila* ya no devuelve el valor 1.
- A partir de la versión 2004 de 4D Internet Commands, Open Transport no se soporta. Por lo tanto, el parámetro *tipoPila* ya no devuelve el valor 2.

El parámetro *versionPila* devuelve en forma de texto el número de versión de la pila TCP utilizada.

IT_Version

IT_Version -> Resultado

Parámetro	Tipo		Descripción
Resultado	Cadena		Cadena de la versión

Descripción

La función *IT_Version* devuelve una cadena de caracteres indicando el número de versión del plug-in 4D Internet Commands.

Ejemplo

El siguiente ejemplo muestra una caja de diálogo de alerta indicando la versión de 4D Internet Commands utilizada.

```
ALERT("4D Internet Commands, versión: "+IT_Version)
```

IT_MacTCPInit

IT_MacTCPInit -> resultado

Parámetro	Tipo		Descripción
resultado	Entero		Código de error

Nota de compatibilidad

El comando *IT_MacTCPInit* ahora no tiene efecto y no debe utilizarse.

IT_MacTCPVer

IT_MacTCPVer (codeVersion) -> resultado

Parámetro	Tipo		Descripción
codeVersion	Entero	⬅	Código de la versión de MacTCP instalada
resultado	Entero	➲	Código de error

Nota de compatibilidad

El comando *IT_MacTCPVer* ha quedado obsoleto con la adición del comando *IT_TCPversion* que permite obtener la información más completa sobre Open Transport y Winsock.

Descripción

A partir de la versión 6.8 de 4D Internet Commands, MacTCP ya no es soportado. Por lo tanto, el parámetro *codeVersion* devuelve 0 sin importar la plataforma y/o sistema operativo.

Anexos

Anexo A: Consejos de programación

Ejecución de los comandos en los bucles Case

En muchos ejemplos de este manual, se utiliza una estructura de programación particular. Muchos de estos ejemplo ejecutan series de comandos que utilizan la instrucción Case de manera poco ortodoxa.

Muchos de los comandos en 4D Internet Commands requieren la ejecución completa de una secuencia de comandos. La falla de un solo comando de la secuencia es suficiente para interrumpir el proceso, el uso de condiciones if en cascada en varios niveles podría ser muy tedioso:

```
If(SMTP_New($smtp_id)=0)
   If(SMTP_Host($smtp_id;◇pref_Server)=0)
      If(SMTP_From($smtp_id;vFrom)=0)
         If(SMTP_To($smtp_id;vTo)=0)
            etc
         End if
      End if
   End if
End if
```

Una alternativa a este método es apoyarse en la forma como 4D ejecuta las instrucciones Case. Cada elemento de una instrucción Case es ejecutado por 4D para determinar si el valor devuelto es **True** o **False**. Si todos los elementos de una instrucción Case devuelven un valor falso, todas las condiciones se ejecutan. Las siguientes líneas reemplazan al código anterior:

```
$SentOK:=False  `Una bandera para indicar si todas las llamadas se enviaron
Case of
   :(SMTP_New($smtp_id)#0)
   :(SMTP_Host($smtp_id;◇pref_Server)#0)
   :(SMTP_From($smtp_id;vFrom)#0)
   :(SMTP_To($smtp_id;vTo)#0)
   :(SMTP_Subject($smtp_id;vSubject)#0)
   :(SMTP_Body($smtp_id;vMessage)#0)
   :(SMTP_Send($smtp_id)#0)
   Else
      $SentOK:=True  `el mensaje se compuso y envío correctamente
End case
If($smtp_id#0)  `Si un mensaje se creó en memoria, se debe borrar ahora
   $OK:=SMTP_Clear($smtp_id)
End if
```

En este ejemplo, cada comando envía el error cero si se completa correctamente. Para poder evaluar cada condición, 4D debe ejecutar cada línea. Como cada condición compara el resultado "diferente de cero", el valor enviado siempre es falso y 4D no encuentra un elemento para detenerse hasta que uno de los comandos falla. Si todos los comandos se ejecutan correctamente, 4D procede con la ejecución del método hasta la condición **Else** donde la bandera $SentOK indica que el mensaje fue compuesto y enviado correctamente.

Sugerencias para una respuesta automática a un correo POP3 o IMAP

Si está pensando en implementar un sistema de correo dentro de su base de datos, donde el usuario puede "responder" los correos recibidos, estas son algunas recomendaciones del RFC # 822:

- La dirección del "remitente" sólo debe enviar respuestas relativas a los problemas en la entrega de correo y no las respuestas relacionadas con el tema del mensaje. Si no existe el campo "Remitente", las notificaciones deben enviarse a la dirección que aparece en el campo "De".

- La dirección del "Remitente" no se debe utilizar en procesos automáticos de respuesta a los mensajes. En cambio, el mensaje debe usar el campo "Responder a" o el campo "De", dependiendo de los factores descritos a continuación.
- Si existe el campo "Responder a" y contiene una o más direcciones de correo, entonces las respuestas deben dirigirse a esta lista. Las direcciones del encabezado "Responder a" tienen prioridad sobre las direcciones listadas en el encabezado "De". Sin embargo, si el campo "Responder a" no existe pero el campo "De" existe, las respuestas deben ser enviadas al buzón de correo indicado en el encabezado "De".

Estas recomendaciones sólo pretenden ayudar en el proceso de decisión, cuando el direccionamiento de correo es manejado por programación en el caso de las acciones de tipo "Respuesta". Una vez creado el mensaje de respuesta, el usuario final puede reemplazar cualquiera de estos valores predeterminados antes de enviar el mensaje.

Anexo B: Números de puertos TCP

Cómo elegir un número de puerto

- 0 a 1023 (Puertos reservados): estos puertos son asignados por la I.A.N.A. (Internet Assigned Numbers Authority) y la mayoría de los sistemas sólo pueden ser utilizados por procesos sistema (o raíz) o por programas ejecutados por los usuarios privilegiados.
  - 20 y 21 FTP;
  - 23 TELNET;
  - 25 SMTP;
  - 37 NTP;
  - 80 y 8080 HTTP;
  - 443 HTTPS.
- 1024 a 49151 (Puertos registrados): estos puertos son listados por la I.A.N.A. y pueden utilizarse en la mayoría de sistemas por procesos usuarios o por los programas ejecutados por los usuarios sin privilegios particulares (routers, aplicaciones especificas...)
- 49152 a 65535 (Puertos dinámicos y/o privados): estos puertos son de uso libre.

Las personas que quieran utilizar los comandos TCP/IP para sincronizar las bases de datos deben utilizar los números de puertos superiores a 49151.

Para mayor información, visite el sitio web de la I.A.N.A: *http://www.iana.org*

Números de puerto TCP

daytime	13	Daytime
qotd	17	Quote of the Day
ftp-data	20	File Transfer [Default Data]
ftp	21	File Transfer [Control]
telnet	23	Telnet
smtp	25	Simple Mail Transfer
time	37	Time
nicname	43	Who Is
domain	53	Domain Name Server
sql*net	66	Oracle SQL*NET
gopher	70	Gopher
finger	79	Finger
http	80	World Wide Web HTTP
poppassd	106	Password Server
rtelnet	107	Remote Telnet Service
pop2	109	Post Office Protocol - Version 2
pop3	110	Post Office Protocol - Version 3
sunrpc	111	SUN Remote Procedure Call
auth	113	Authentication Service
sftp	115	Simple File Transfer Protocol
sqlserv	118	SQL Services
nntp	119	Network News Transfer Protocol
ntp	123	Network Time Protocol
pwdgen	129	Password Generator Protocol
imap2	143	Interactive Mail Access Protocol v2
news	144	NewS
sql-net	150	SQL-NET
multiplex	171	Network Innovations Multiplex
cl/1	172	Network Innovations CL/1
at-rtmp	201	AppleTalk Routing Maintenance
at-nbp	202	AppleTalk Name Binding
at-3	203	AppleTalk Unused
at-echo	204	AppleTalk Echo
at-5	205	AppleTalk Unused
at-zis	206	AppleTalk Zone Information
at-7	207	AppleTalk Unused
at-8	208	AppleTalk Unused
ipx	213	IPX
netware-ip	396	Novell Netware over IP
timbuktu	407	Timbuktu
https	443	Secured protocol
conference	531	chat
netnews	532	readnews
netwall	533	for emergency broadcasts
uucp	540	uucpd
uucp-rlogin	541	uucp-rlogin
whoami	565	whoami
ipcserver	600	Sun IPC server
phonebook	767	phone
accessbuilder	888	AccessBuilder

Anexo C, Códigos de error de 4D Internet Commands

Todos 4D Internet Commands (con excepción de *IT_ErrorText* y *IT_Version*) devuelven un valor entero como resultado. Este entero contiene un número de error que el comando debe devolver a la base de datos 4D. Si un comando se ejecuta correctamente, devuelve cero.

El origen de un error puede generalmente determinarse a partir del rango de valores en el que se sitúa su número. La siguiente tabla ofrece un índice a las principales fuentes de error en función de su número:

Número de error	**Generado por**
Error < 0	Sistema operativo o capa de red WinSock
0	Sin error
Error 1 -> 61	capa de red BSD
Error >= 10000	4D Internet Commands

Códigos de errores 4D Internet Commands

Si un error se produce durante la ejecución de un comando Internet de 4D, se devuelve uno de los siguientes valores:

10000 user cancelled a dialog or progress.
10001 unimplemented Internet command.
10002 invalid array type.
10003 no more (TCP,SMTP,POP3, etc.) references available.
10004 invalid reference.
10005 need a "Host" for use in the "SMTP_Send" command.
10006 need a "From" for use in the "SMTP_Send" command.
10007 need a recipient for use in the "SMTP_Send" command.
10008 already logged in.
10009 error trying to make a POP3 connection.
10010 error with POP3 USER.
10011 error with POP3 PASS.
10012 error with POP3 QUIT.
10013 error with POP3 STAT.
10014 error with POP3 LIST.
10015 error with POP3 UIDL.
10016 error with POP3 DELE.
10017 error with POP3 RSET.
10018 invalid message number.
10019 invalid character offset.
10020 invalid character length.
10021 error with POP3 RETR.
10022 field was not found in mail Header.
10023 no attachments found.
10024 error in processing BinHex.
10025 BinHex checksum error.
10026 Internet commands unavailable. Probably because MacTCP is not installed
10027 Connection no longer exists
10028 Exceeded 32k limit
10029 Error with POP3 NOOP
10030 POP3 session was closed by the server
10031 Error with POP3 APOP
10032 Unknown or invalid response.
10033 SMTP 421 - Service not available, closing transmission channel.
10034 SMTP 450 - Requested mail action not taken: mailbox unavailable.
10035 SMTP 451 - Requested action aborted: local error in processing.
10036 SMTP 452 - Requested action not taken: insufficient system storage.
10037 SMTP 500 - Syntax error, command unrecognized.
10038 SMTP 501 - Syntax error in parameters or arguments.
10039 SMTP 502 - Command not implemented.
10040 SMTP 503 - Bad sequence of commands.
10041 SMTP 504 - Command parameter not implemented.
10042 SMTP 550 - Requested action not taken: mailbox unavailable.
10043 SMTP 551 - User not local; please try <forward-path>.
10044 SMTP 552 - Requested mail action aborted: exceeded storage allocation.
10045 SMTP 553 - Requested action not taken: mailbox name not allowed.
10046 SMTP 554 - Transaction failed.
10047 FTP 421 - Service not available, closing control connection.
10048 FTP 425 - Can't open data connection.
10049 FTP 426 - Connection closed; transfer aborted.
10050 FTP 450 - Requested file action not taken. File unavailable (e.g.,file busy).
10051 FTP 451 - Requested action aborted: local error in processing.
10052 FTP 452 - Requested action not taken. Insufficient storage space in system.
10053 FTP 500 - Syntax error, command unrecognized.
10054 FTP 501 - Syntax error in parameters or arguments.

10055	FTP 502 - Command not implemented.
10056	FTP 503 - Bad sequence of commands.
10057	FTP 504 - Command not implemented for that parameter.
10058	FTP 530 - Not logged in.
10059	FTP 532 - Need account for storing files.
10060	FTP 550 - Requested action not taken. File unavailable (e.g., file not found, no access).
10061	FTP 551 - Requested action aborted: page type unknown.
10062	FTP 552 - Requested file action aborted. Exceeded storage allocation (for current directory or dataset).
10063	FTP 553 - Requested action not taken. File name not allowed.
10064	No response has been received within the given timeout period.
10065	Not an FTP file.
10066	Error in processing Base64.
10067	Error in processing AppleSingle.
10068	Error in processing Quoted-Printable.
10069	FTP session was closed by the server.
10070	Not an FTP directory.
10071	TCP session was closed by the server
10072	Invalid encode kind
10073	Invalid decode kind
10074	An asynchronous DNR call did not complete
10075	An asynchronous OpenTransport call did not complete
10076	OpenTransport bind failed
10077	OpenTransport connect failed
10078	Maximum MacTCP streams reached
10079	Error in processing uuencode
10080	Cannot load ICMP library
10081	Error in processing MacBinary
10082	MacBinary checksum error
10083	Could not open a file
10084	No FTP information received
10085	Unknown FTP information received
10086	Proxy connection failed
10087	Standard file I/O error
10088	FTP reentrant error
10089	SLI.DLL is not loaded
10091	Error trying to make an IMAP connection
10092	A maibox is not selected
10093	Invalid message part
10094	Error with IMAP LOGIN
10095	Error with IMAP LOGOUT
10096	Error with IMAP CAPABILITY
10097	Error with IMAP SELECT
10098	Error with IMAP FETCH
10099	Error with IMAP PARTIAL
10100	Error with IMAP STORE
10101	Error with IMAP EXPUNGE
10102	Error with IMAP SEARCH
10103	Error with IMAP COPY
10104	Error with IMAP CREATE
10105	Error with IMAP DELETE
10106	Error with IMAP RENAME
10107	Error with IMAP SUBSCRIBE
10108	Error with IMAP UNSUBSCRIBE
10109	Error with IMAP LIST

10110	Error with IMAP LSUB
10111	Error with IMAP STATUS
10112	Error with IMAP CLOSE
10113	Error with AUTHENTICATION

Códigos de errores BSD

1	Operation not permitted
4	Interrupted system call
13	Permission denied
14	Bad address
22	Invalid argument
24	Too many open files
35	Operation would block
36	Operation now in progress
37	Operation already in progress
38	Socket operation on non-socket
39	Destination address required
40	Message too long
41	Protocol wrong type for socket
42	Protocol not available
43	Protocol not supported
44	Socket type not supported
45	Operation not supported
46	Protocol family not supported
47	Address family not supported by protocol family
48	Address already in use
49	Can't assign requested address
50	Network is down
51	Network is unreachable
52	Network dropped connection on reset
53	Software caused connection abort
54	Connection reset by peer
55	No buffer space available
56	Socket is already connected
57	Socket is not connected
58	Can't send after socket shutdown
60	Operation timed out
61	Connection refused

Código de errores WinSock

-10004	Blocking call cancelled
-10013	Permission denied
-10014	Bad address
-10022	Invalid argument
-10024	No more sockets available
-10035	Non-blocking socket would block
-10036	Illegal WinSock function invoked while a blocking function is in progress
-10037	An attempt was made to cancel an asynchronous operation that has already completed
-10038	Specified socket descriptor is not valid for this application
-10039	Destination address was required but none was supplied to the function
-10040	Datagram too large for buffer
-10041	Specified protocol does not match the other parameters in the call
-10042	Protocol option is unknown or invalid
-10043	Specified protocol is not supported by the Windows Sockets implementation
-10044	Specified socket type is not supported by the specified address family
-10045	Socket does not support the specified operation
-10046	Protocol family not supported
-10047	Specified address family is not supported by the Windows Sockets implementation or cannot be used with the indicated socket
-10048	Specified address is already in use
-10049	Specified address is not available from the local machine
-10050	Problem with the network subsystem
-10051	Network cannot be reached from this host at this time
-10052	Connection was dropped and must be reset
-10053	Connection was aborted because of a timeout or other error condition
-10054	Connection was reset by the remote host
-10055	Windows Sockets implementation is out of buffer space or the space provided in an API call by the application was too small to hold the requested information
-10056	Specified socket is already connected
-10057	Specified socket is not connected
-10058	Socket has had the requested functionality shut down
-10060	Connection attempt timed out before the connection could be established
-10061	Connection attempt was forcefully rejected
-10091	Network subsystem is not yet ready for communication
-10092	Windows Sockets DLL does not support the requested Winsock protocol version
-10093	Windows Sockets not initialized
-11001	Requested database information does not exist; as confirmed by an authoritative host
-11002	Requested information was not found but the answer was not authoritative
-11003	Non-recoverable error occurred
-11004	Name supplied was valid but no information of the requested type is in the database

Codificación SMTP

Los siguientes valores **no** son códigos de error devueltos por los comandos Internet de 4D. Estos son los códigos de respuestas definidos para el protocolo SMTP con el fin de indicar el estado actual de la comunicación en modo cliente-servidor. Esta lista es particularmente útil cuando crea su propio sistema de mensajería con la ayuda de los comandos TCP de bajo nivel.

211	System status, or system help reply
214	Help message [Information on how to use the receiver or the meaning of a particular non-standard command; this reply is useful only to the human user]
220	<domain> Service ready
221	<domain> Service closing transmission channel
250	Requested mail action okay, completed
251	User not local; will forward to <forward-path>
354	Start mail input; end with <CRLF>.<CRLF>
421	<domain> Service not available, closing transmission channel [This may be a reply to any command if the service knows it must shut down]
450	Requested mail action not taken: mailbox unavailable [e.g., mailbox busy]
451	Requested action aborted: local error in processing
452	Requested action not taken: insufficient system storage
500	Syntax error, command unrecognized [This may include errors such as command line too long]
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command parameter not implemented
550	Requested action not taken: mailbox unavailable [e.g., mailbox not found, no access]
551	User not local; please try <forward-path>
552	Requested mail action aborted: exceeded storage allocation
553	Requested action not taken: mailbox name not allowed [e.g., mailbox syntax incorrect]
554	Transaction failed

Codificación FTP

Los siguientes valores **no** son códigos de error devueltos por los comandos Internet de 4D. Estos son los códigos de respuestas definidos para el protocolo FTP con el fin de indicar el estado actual de la comunicación en modo cliente-servidor. Esta lista es particularmente útil cuando crea su propio sistema de mensajería con la ayuda de los comandos FTP de bajo nivel.

110	Restart marker reply. In this case, the text is exact and not left to the particular implementation; it must read: MARK yyyy = mmmm. Where yyyy is User-process data stream marker, and mmmm server's equivalent marker (note the spaces between markers and "=").
120	Service ready in nnn minutes.
125	Data connection already open; transfer starting.
150	File status okay; about to open data connection.
200	Command okay.
202	Command not implemented, superfluous at this site.
211	System status, or system help reply.
212	Directory status.
213	File status.
214	Help message on how to use the server or the meaning of a particular non-standard command. This reply is useful only to the human user.
215	NAME system type. Where NAME is an official system name from the list in the Assigned Numbers document.
220	Service ready for new user.
221	Service closing control connection. Logged out if appropriate.
225	Data connection open; no transfer in progress.
226	Closing data connection. Requested file action successful (e.g., file transfer or file abort).
227	Entering Passive Mode (h1,h2,h3,h4,p1,p2).
230	User logged in, proceed.
250	Requested file action okay, completed.
257	"PATHNAME" created.
331	User name okay, need password.
332	Need account for login.
350	Requested file action pending further information.
421	Service not available, closing control connection. This may be a reply to any command if the service knows it must shut down.
425	Can't open data connection.
426	Connection closed; transfer aborted.
450	Requested file action not taken. File unavailable (file busy).
451	Requested action aborted: local error in processing.
452	Requested action not taken. Insufficient storage space in system.
500	Syntax error, command unrecognized. This may include errors such as command line too long.
501	Syntax error in parameters or arguments.
502	Command not implemented.
503	Bad sequence of commands.
504	Command not implemented for that parameter.
530	Not logged in.
532	Need account for storing files.
550	Requested action not taken. File unavailable (e.g., file not found, no access).
551	Requested action aborted: page type unknown.
552	Requested file action aborted. Exceeded storage allocation (for current directory or dataset).
553	Requested action not taken. File name not allowed.

Anexo D: Información adicional

Las siguientes referencias son los URLs adicionales relacionados con los protocolos Internet.

http://www.internic.net/: para entender qué es un nombre de dominio y lo que debe hacer para registrar uno.

http://www.ietf.org/: sitio de Internet Engineering Task Force (IETF).

http://www.rfc-editor.org/: para entender qué es un RFC y saber cómo buscarlos (http://www.rfc-editor.org/rfc.html).

ftp://ftp.isi.edu/in-notes/rfc821.txt: Simple Mail Transfer Protocol -- RFC 821.

http://www.w3c.org/: todo lo que debe saber sobre la World Wide Web.

http://www.imap.org: sitio dedicado al protocolo IMAP.

Appendix E, Mocking network to file

Testing and debugging code that sends emails could be a pain; when an email is not received correctly, the source of the problem could be the network configuration, the provider, the client software, and so on.

To help you in this case, we added the ability to dump email in a local file instead of sending it. With this feature, you just need to modify the resulting file slighlty to create an EML file that can display the results in MS Outlook. You can also include email files in unit-testing procedures.

Code for the local test

You can run the local code:

```
$err:=SMTP_SetPrefs(0;15;0) // Body: UTF-8 & QuotedPrintable, Header: UTF-8 & Base64
$err:=SMTP_Charset(1;1) // Apply setting to header and body (Body: UTF-8 & QuotedPrintable & Header: UTF-8 & Base64)

//$hostName:="smtp.gmail.com" // This line sends it over the network using smtp.gmail.com
$hostName:="file:C:\\Users\\MyWinUser\\Desktop\\test.txt" // This line doesn't send the email, instead it saves the bytes in a file you can monitor
$msgTo:="mail.to@gmail.com"
$msgFrom:="mail.sender@gmail.com"

$mailSubject:="テストメール(v17 4372) " //test using extended characters
$mailBody:="日本語で終わる"
$err:=SMTP_QuickSend($hostName;$msgFrom;$msgTo;$mailSubject;$mailBody;0;0;$msgFrom;"password")
```

This produces the following .txt file:

```
<mail.sender@gmail.com>
<mail.to@gmail.com>

Mime-Version: 1.0
Content-Type: text/plain;charset="utf-8"
Content-Transfer-Encoding: quoted-printable
Date: Fri, 08 Jul 2016 16:45:24 +0200
To: mail.to@gmail.com
From: mail.sender@gmail.com
Subject: =?utf-8?B?44OG44K544OI44Oh44O844Or77yIdjE3IDQzNzIpIA==?=

=E6=97=A5=E6=9C=AC=E8=AA=9E=E3=81=A7=E7=B5=82=E3=82=8F=E3=82=8B
```

If you want to open this file as a standard MS Outlook email:

1. Delete all lines before "Mime-Version: 1.0":

```
Mime-Version: 1.0
Content-Type: text/plain;charset="utf-8"
Content-Transfer-Encoding: quoted-printable
Date: Fri, 08 Jul 2016 16:45:24 +0200
To: mail.to@gmail.com
From: mail.sender@gmail.com
```

```
Subject: =?utf-8?B?44OG44K544OI44Oh44O844Or77yIdjE3IDQzNzIpIA==?=

=E6=97=A5=E6=9C=AC=E8=AA=9E=E3=81=A7=E7=B5=82=E3=82=8F=E3=82=8B
```

2. Save this file with the ".eml" extension, for example "test.eml".
3. Double-click the file and you will see the email in MS Outlook just as if you received it from your mail server.

Timestamp considerations

To comply with unit-testing mechanisms, when using **SMTP_QuickSend** with an output file, the Date header is always the following string:

```
Date: Fri, 08 Jul 2016 16:45:24 +0200
```

In this case, date comparisons will not fail in unit tests.

Note: When you use a real host (such as smtp.gmail.com), the date header gets replaced by an actual timestamp.

If you want to have an actual timestamp in your test file, you can use this feature with **SMTP_Send**. In this case, you can use the **SMTP_Date** command and then provide an actual Date header with **SMTP_Send**.