

# Instalación

¡Bienvenido a 4D! ¡Bienvenido a 4D! A continuación encontrará toda la información necesaria sobre cómo instalar y registrar su aplicación 4D. En esta página encontrará toda la información necesaria para instalar y lanzar su producto 4D.

## Configuración requerida

La página [Descarga de productos](#) del sitio web de 4D ofrece información sobre los requisitos mínimos del sistema macOS / Windows para su serie 4D.

Encontrará más detalles técnicos en la [página Recursos](#) del sitio web de 4D.

## Instalación en disco

Los productos 4D se instalan desde el sitio web de 4D:

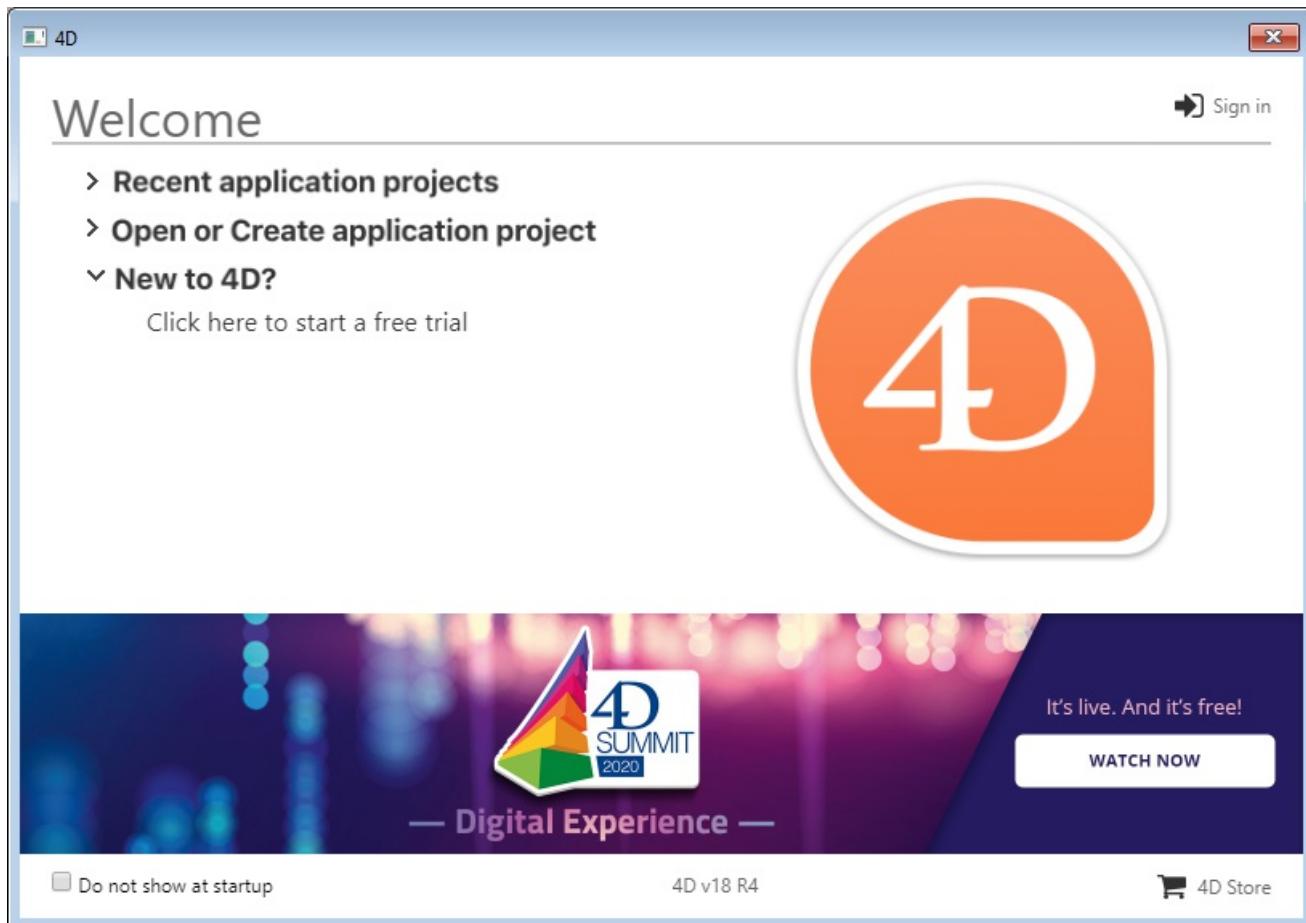
1. Conéctese al sitio web de 4D y vaya a la página de [Descargas](#).
2. Haga clic en el enlace de descarga de su producto 4D y siga las instrucciones en pantalla.

## Conexión

Una vez que haya completado la instalación, puede iniciar 4D e iniciar la sesión. Para ello, haga doble clic en el icono del producto 4D.



A continuación, aparece el Asistente de bienvenida:



- Si desea descubrir y explorar 4D, haga clic en el enlace prueba gratuita. Sólo se le pedirá que se registre o que cree una cuenta 4D.
- Si ya tiene una cuenta en 4D, haga clic en el enlace Iniciar sesión en la parte superior derecha del diálogo del Asistente de Bienvenida e introduzca los datos de su cuenta. Toda licencia 4D ya registrada se actualiza automáticamente (o se cargan paquetes de expansión adicionales) en su máquina.

Despliegue el área Abrir o crear un proyecto aplicación y seleccione la acción que desea realizar:

- Conéctese a 4D Server - utilice 4D como cliente remoto y conéctese a una aplicación ya cargada por 4D Server.
- Abra un proyecto de aplicación local: cargue un proyecto de aplicación existente almacenado en su disco.
- Cree un nuevo proyecto de aplicación : cree un nuevo proyecto de aplicación vacío en su disco.

¡Disfrute de su experiencia 4D!

# Arquitectura de un proyecto

Un proyecto 4D se compone de varias carpetas y archivos, almacenados dentro de una única carpeta padre de la aplicación (carpeta paquete). Por ejemplo:

- MyProject
  - Componentes
  - Datos
    - Logs
    - Settings
  - Documentation
  - Plugins
  - Project
    - DerivedData
    - Sources
    - Trash
  - Resources
  - Settings
  - userPreferences.jSmith
  - WebFolder

Si su proyecto se ha convertido desde una base binaria, puede haber carpetas adicionales. Ver "Conversión de bases en proyectos" en [doc.4d.com](#).

## Carpeta Project

La carpeta Project suele contener la siguiente jerarquía:

- archivo `<applicationName>.4DProject`
- `Sources`
  - Clases
  - DatabaseMethods
  - Métodos
  - Formularios
  - TableForms
  - Triggers
- `DerivedData`
- `Trash` (si hay)

### archivo `<applicationName>.4DProject`

El archivo de desarrollo de proyecto, utilizado para designar y lanzar el proyecto. Este archivo puede ser abierto por:

- 4D
- 4D Server (sólo lectura, ver [Abrir un proyecto remoto](#))

En los proyectos 4D, el desarrollo se realiza con 4D y el desarrollo multiusuarios se gestiona a través de las herramientas de control de versión. 4D Server puede abrir archivos .4DProject para realizar pruebas.

Este archivo texto también puede contener llaves de configuración, en particular `"tokenizedText": false`.

## Sources

Contenido	Descripción	Formato
catalog.4DCatalog	Definiciones de tablas y campos	XML
folders.json	Definiciones de carpetas del Explorador	JSON
menus.json	Definiciones de los menús	JSON
settings.4DSettings	Propiedades de la base <i>Structure</i> . No se tienen en cuenta si los <i>parámetros usuario</i> o los <i>parámetros usuario de datos</i> son definidos. Atención: en las aplicaciones compiladas, la configuración de la estructura se almacena en el archivo .4dz (de sólo lectura). Para las necesidades de despliegue, es necesario utilizar los <i>parámetros usuario</i> o los <i>parámetros usuario para los datos</i> para definir la configuración personalizada.	XML
tips.json	Mensajes de ayuda definidos	JSON
lists.json	Listas definidas	JSON
filters.json	Filtros definidos	JSON
styleSheets.css	Hojas de estilo CSS	CSS
styleSheets_mac.css	Hojas de estilo css de Mac (a partir de una base binaria convertida)	CSS
styleSheets_windows.css	Hojas de estilo css en Windows (a partir de una base binaria convertida)	CSS

## DatabaseMethods

Contenido	Descripción	Formato
databaseMethodName.4dm	Métodos base definidos en el proyecto. Un archivo por método base	texto

## Métodos

Contenido	Descripción	Formato
methodName.4dm	Métodos proyecto definidos en el proyecto. Un archivo por método	texto

## Clases

Contenido	Descripción	Formato
className.4dm	Método de definición de clases usuario, que permite instanciar objetos específicos. Un archivo por clase, el nombre del archivo es el nombre de la clase	texto

## Formularios

Contenido	Descripción	Formato
formName/form.4DForm	Descripción del formulario proyecto	json
formName/method.4dm	Método formulario proyecto	texto
formName/Images/pictureName	Imagen estática del formulario proyecto	imagen
formName/ObjectMethods/objectName.4dm	Métodos objeto. Un archivo por método objeto	texto

## TableForms

Contenido	Descripción	Formato
<i>n/Input/formName/form.4DForm</i>	Descripción del formulario de entrada de la tabla (n es el número de tabla)	json
<i>n/Input/formName/Images/pictureName</i>	Imágenes estáticas del formulario de entrada de la tabla	imagen
<i>n/Input/formName/method.4dm</i>	Método del formulario de entrada de la tabla	texto
<i>n/Input/formName/ObjectMethods/ObjectName.4dm</i>	Métodos objeto del formulario de entrada. Un archivo por método objeto	texto
<i>n/Output/formName/form.4DForm</i>	Descripción del formulario de salida de la tabla (n es el número de tabla)	json
<i>n/Output/formName/Images/pictureName</i>	Imágenes estáticas del formulario de salida de la tabla	imagen
<i>n/Output/formName/method.4dm</i>	Método del formulario de salida de la tabla	texto
<i>n/Output/formName/ObjectMethods/ObjectName.4dm</i>	Métodos objeto del formulario de salida. Un archivo por método objeto	texto

## Triggers

Contenido	Descripción	Formato
<i>table_n.4dm</i>	Métodos trigger definidos en el proyecto. Un archivo de activación por tabla (n es el número de tabla)	texto

Nota: La extensión de archivo .4dm es un formato de archivo texto, que contiene el código de un método 4D. Es compatible con las herramientas de control de versión.

## Trash

La carpeta Trash contiene los métodos y formularios que se han eliminado del proyecto (si los hay). Puede contener las siguientes carpetas:

- [Métodos](#)
- [Formularios](#)
- [TableForms](#)

Dentro de estas carpetas, los nombres de los elementos eliminados van entre paréntesis, por ejemplo "*(myMethod).4dm*". La organización de las carpetas es idéntica a la carpeta [Sources](#).

## DerivedData

La carpeta DerivedData contiene datos en caché utilizados internamente por 4D para optimizar el procesamiento. Se crea o recrea automáticamente cuando es necesario. Puede ignorar esta carpeta.

## Libraries

Esta carpeta se utiliza sólo en macOS.

La carpeta Librairies contiene el archivo resultante de una compilación con el [compilador Silicon](#) en macOS.

## Resources

La carpeta Resources contiene todos los archivos y carpetas de recursos personalizados del proyecto. En esta carpeta puede colocar todos los archivos necesarios para la traducción o personalización de la interfaz de la aplicación (archivos

imagen, archivos texto, archivos XLIFF, etc.). 4D utiliza mecanismos automáticos para trabajar con el contenido de esta carpeta, en particular para el manejo de archivos XLIFF e imágenes estáticas. Para su uso en modo remoto, la carpeta Resources permite compartir archivos entre el equipo servidor y todos los equipos cliente. Ver el *manual 4D Server*.

Contenido	Descripción	Formato
item	Archivos y carpetas de recursos del proyecto	varios
Images/Library/item	Imágenes de la librería de imágenes como archivos separados(*). Los nombres de estos elementos se convierten en nombres de archivos. Si existe un duplicado, se añade un número al nombre.	imagen

(\*) sólo si el proyecto fue exportado desde una base binaria .4db.

## Datos

La carpeta Data contiene el archivo de datos y todos los archivos y carpetas relacionados con los datos.

Contenido	Descripción	Formato
data.4dd(*)	Archivo de datos que contiene los datos introducidos en los registros y todos los datos pertenecientes a los registros. Al abrir un proyecto 4D, la aplicación abre por defecto el archivo de datos actual. Si cambia el nombre o la ubicación de este archivo, aparecerá la caja de diálogo <i>Abrir un archivo de datos</i> para que pueda seleccionar el archivo de datos a utilizar o crear uno nuevo	binario
data.journal	Se crea sólo cuando la base de datos utiliza un archivo de registro. El archivo de registro se utiliza para garantizar la seguridad de los datos entre las copias de seguridad. Todas las operaciones realizadas sobre los datos se registran secuencialmente en este archivo. Por lo tanto, cada operación sobre los datos provoca dos acciones simultáneas: la primera sobre los datos (la instrucción se ejecuta normalmente) y la segunda en el archivo de registro (se registra una descripción de la operación). El archivo de registro se construye de forma independiente, sin perturbar ni ralentizar el trabajo del usuario. Una base de datos sólo puede trabajar con un único archivo de registro a la vez. El archivo de historial registra operaciones como adiciones, modificaciones o eliminaciones de registros, transacciones, etc. Se genera por defecto cuando se crea una base de datos.	binario
data.match	(interno) UUID correspondiente al número de la tabla	XML

(\*) Cuando el proyecto se crea a partir de una base de datos binaria .4db, el archivo de datos se deja intacto. Por lo tanto, se puede nombrar de otra manera y colocar en otro lugar.

## Settings

Esta carpeta contiene archivos de configuración de datos utilizados para la administración de la aplicación.

Estos parámetros tienen prioridad sobre los [archivos de propiedades usuario](#) y los [archivos de propiedades estructura](#).

Contenido	Descripción	Formato
directory.json	Descripción de los grupos y usuarios de 4D y sus derechos de acceso cuando la aplicación se lanza con este archivo de datos.	JSON
Backup.4DSettings	Parámetros de copia de seguridad de la base de datos, utilizados para definir las <a href="#">opciones de copia de seguridad</a> cuando la base se lanza con este archivo de datos. Las llaves relativas a la configuración de la copia de seguridad se describen en el manual <i>Backup de las llaves XML 4D</i> .	XML
settings.4DSettings	Propiedades de la base personalizadas para este archivo de datos.	XML

## Logs

La carpeta Logs contiene todos los archivos de registro utilizados por el proyecto. Los archivos de registro incluyen, en particular:

- conversión de base de datos,
- peticiones del servidor web,
- registro de actividades de backup/restitución (*Backup Journal/[xxx].txt*, ver [Historial de backup](#))
- depuración de comandos,
- Peticiones 4D Server (generadas en los equipos cliente y en el servidor).

Una carpeta Logs adicional está disponible en la carpeta de preferencias del usuario del sistema (carpeta 4D activa, ver el comando [Get 4D folder](#)) para los archivos de registro de mantenimiento y en los casos en que la carpeta de datos es de sólo lectura.

## Settings

Esta carpeta contiene archivos de propiedades usuario utilizados para la administración de la aplicación.

Estos parámetros tienen prioridad sobre los archivos [propiedades estructura](#). Sin embargo, si existe un [archivo de parámetros usuario](#), éste tiene prioridad sobre el archivo de las propiedades usuario.

Contenido	Descripción	Formato
directory.json	Descripción de los grupos y usuarios de 4D para la aplicación, así como sus derechos de acceso	JSON
Backup.4DSettings	Parámetros de copia de seguridad de la base de datos, utilizados para definir las <a href="#">opciones de copia de seguridad</a> cuando se lanza cada copia de seguridad. Este archivo también puede utilizarse para leer o definir opciones adicionales, como la cantidad de información almacenada en el <i>diario de backup</i> . Las llaves relativas a la configuración de la copia de seguridad se describen en el manual <i>Backup de las llaves XML 4D</i> .	XML
BuildApp.4DSettings	Archivo de parámetros de generación, creado automáticamente cuando se utiliza la caja de diálogo del generador de aplicaciones o del comando <code>BUILD APPLICATION</code>	XML
settings.4DSettings	Parámetros personalizados para este proyecto (todos los archivos de datos)	XML

## userPreferences.<userName>

Esta carpeta contiene archivos que memorizan las configuraciones del usuario, por ejemplo, el punto de ruptura o las posiciones de las ventanas. Puede simplemente ignorar esta carpeta. Contiene, por ejemplo:

Contenido	Descripción	Formato
methodPreferences.json	Preferencias del editor de métodos del usuario actual	JSON
methodWindowPositions.json	Posición de la ventana de usuario actual para los métodos	JSON
formWindowPositions.json	Posición de la ventana de usuario actual para los formularios	JSON
workspace.json	Lista de ventanas abiertas; en macOS, orden de las ventanas de la pestaña	JSON
debuggerCatches.json	Llamadas a los comandos	JSON
recentTables.json	Lista ordenada de tablas	JSON
preferences.4DPreferences	Ruta de datos actual y posiciones de la ventana principal	XML
CompilerIntermediateFiles	Archivos intermedios resultantes de la compilación Apple Silicon	Folder

## Componentes

Esta carpeta contiene los componentes que deben estar disponibles en el proyecto aplicación. Debe almacenarse en el mismo nivel que la carpeta Project.

Una aplicación proyecto puede ser utilizada por sí misma como un componente: - para el desarrollo: inserte un alias del archivo .4dproject en la carpeta Components del proyecto local. - para el despliegue: [cree el componente](#) y coloque el archivo .4dz resultante en una carpeta .4dbase en la carpeta Components de la aplicación local.

## Plugins

Esta carpeta contiene los plug-ins que deben estar disponibles en el proyecto aplicación. Debe almacenarse en el mismo nivel que la carpeta Project.

## Documentation

Esta carpeta contiene todos los archivos de documentación (.md) creados para los elementos del proyecto como clases, métodos o formularios. Los archivos de documentación se gestionan y se muestran en el Explorador 4D.

Para más información, consulte [Documentar un proyecto](#).

## WebFolder

Define la carpeta raíz por defecto del servidor web 4D para las páginas, las imágenes, etc. Se crea automáticamente cuando se lanza el servidor web por primera vez.

## Archivo `.gitignore` (opcional)

Archivo que especifica qué archivos serán ignorados por git. Puede incluir un archivo gitignore en sus proyectos utilizando la opción Crear un archivo .gitignore en la página General de las preferencias. Para configurar el contenido de ese archivo, ver [Crear un archivo .gitignore](#).

# Documentar un proyecto

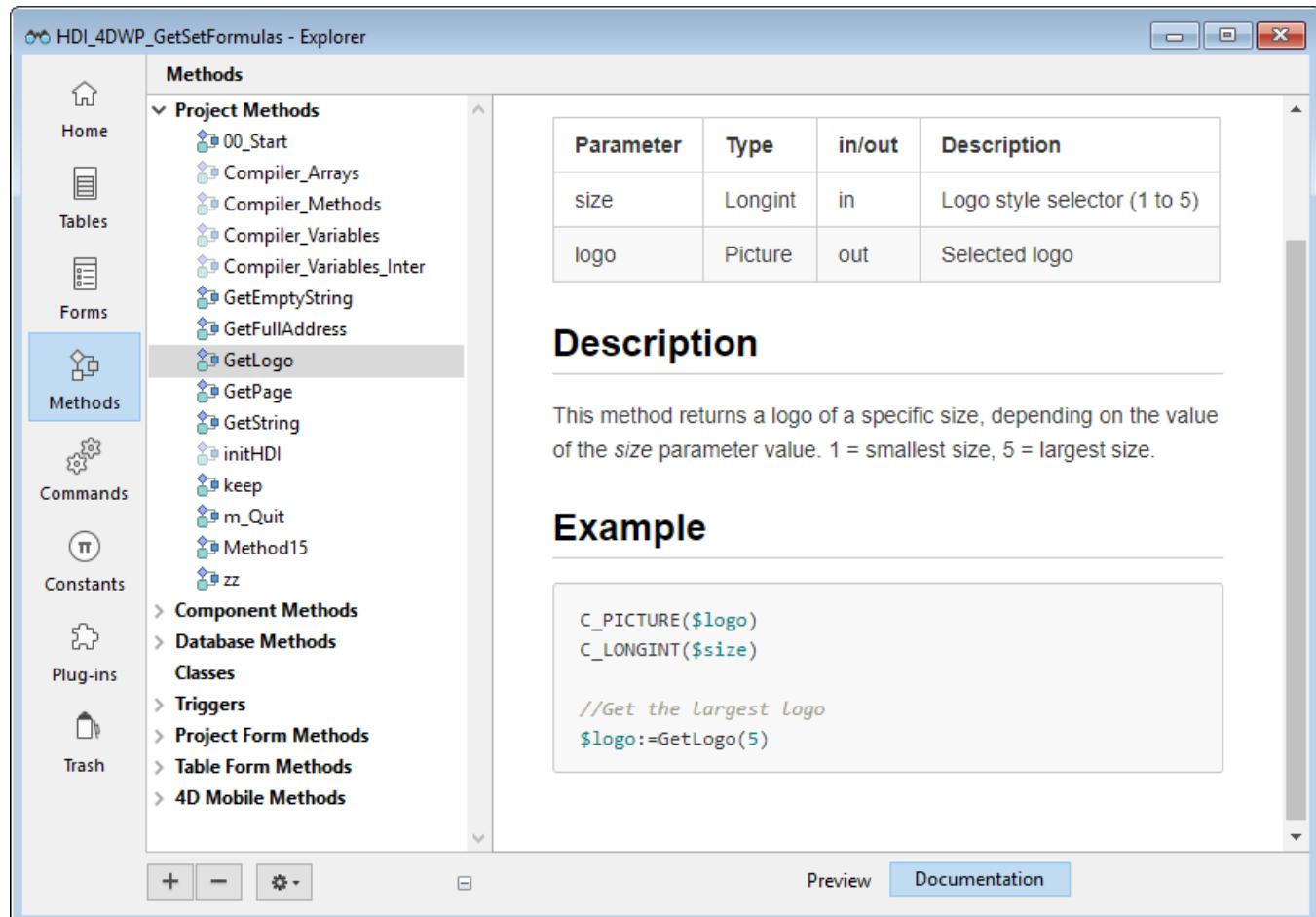
En los proyectos aplicación, puede documentar sus métodos así como sus formularios, tablas o campos. La creación de documentación es especialmente apropiada para proyectos desarrollados por varios programadores y, en general, es una buena práctica de programación. La documentación puede contener una descripción de un elemento, así como cualquier información necesaria para entender cómo funciona el elemento en la aplicación.

Los siguientes elementos del proyecto aceptan la documentación:

- Métodos (métodos base, métodos componente, métodos proyecto, métodos formulario, métodos 4D Mobile, triggers y clases)
- Formularios
- Tablas y campos

Sus archivos de documentación se escriben en la sintaxis Markdown (archivos .md) utilizando cualquier editor que soporte el Markdown. Se almacenan como archivos independientes dentro de la carpeta Proyecto.

La documentación se muestra en el área de vista previa (panel lateral derecho) del Explorador:



También se puede exponer parcialmente como [consejos del editor de código](#).

## Archivos documentación

### Nombre del archivo de documentación

Los archivos de documentación tienen el mismo nombre que su elemento adjunto, con la extensión ".md". Por ejemplo, el archivo de documentación adjunto al método proyecto `myMethod.4dm` se llamará `myMethod.md`.

En el Explorador, 4D muestra automáticamente el archivo de documentación con el mismo nombre que el elemento seleccionado (ver abajo).

## Arquitectura de los archivos de documentación

Todos los archivos de documentación se almacenan en la carpeta `Documentación`, situada en el primer nivel de la carpeta Package.

La arquitectura de la carpeta `Documentation` es la siguiente:

- `Documentation`
  - `Clases`
    - `myClass.md`
  - `DatabaseMethods`
    - `onStartup.md`
    - ...
  - `Formularios`
    - `loginDial.md`
    - ...
  - `Métodos`
    - `myMethod.md`
    - ...
  - `TableForms`
    - `1`
      - `input.md`
      - ...
    - ...
  - `Triggers`
    - `table1.md`
    - ...
- Un formulario proyecto y su método de formulario proyecto comparten el mismo archivo de documentación para el formulario y el método.
- Un formulario tabla y su método de formulario tabla comparten el mismo archivo de documentación para el formulario y el método.

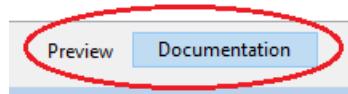
Renombrar o eliminar un elemento documentado en su proyecto también renombrará o eliminará el archivo Markdown asociado al elemento.

## Documentación en el Explorador

### Ver la documentación

Para ver la documentación en la ventana del Explorador:

1. Asegúrese de que se muestra el área de vista previa.
2. Seleccione el elemento documentado en la lista del Explorador.
3. Haga clic en el botón `Documentation` situado debajo del área de vista previa.



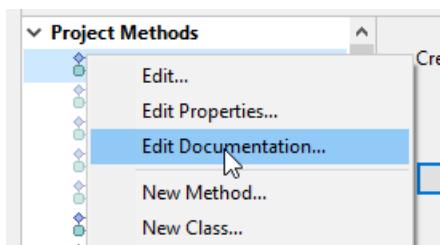
- Si no se ha encontrado ningún archivo de documentación para el elemento seleccionado, se muestra un botón `Crear` (ver más abajo).
- De lo contrario, si existe un archivo de documentación para el elemento seleccionado, el contenido se muestra en el área. El contenido no se puede editar directamente en el panel.

### Modificar el archivo de documentación

Puede crear y/o editar un archivo de documentación Markdown desde la ventana del Explorador para el elemento seleccionado.

Si no existe un archivo de documentación para el elemento seleccionado, puede:

- haga clic en el botón Crear en el panel Documentation o,
- elija la opción Modificar la documentación... en el menú contextual o el menú de opciones del Explorador.

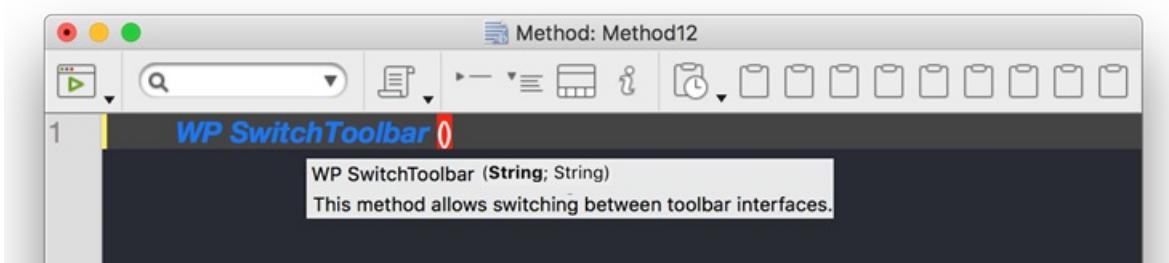


4D crea automáticamente un archivo .md con el nombre adecuado y una plantilla básica en la ubicación correspondiente y lo abre con su editor Markdown predeterminado.

Si ya existe un archivo de documentación para el elemento seleccionado, puede abrirlo con su editor de Markdown eligiendo la opción Modificar la documentación... del menú contextual o del menú de opciones del Explorador.

## Visualizar la documentación en el editor de código

El editor de código 4D muestra una parte de la documentación de un método en su consejo de ayuda.



Si un archivo llamado "<MethodName>.md" existe en la carpeta "<package>/documentation", el editor de código muestra (por prioridad):

- Todo texto introducido en una etiqueta de `comentario HTML (<!-- comando documentation --> )` en la parte superior del archivo markdown.
  - O, si no se utiliza la etiqueta de `comentario html`, la primera frase después de una etiqueta `# Description` del archivo markdown.
- En este caso, la primera línea contiene el prototipo del método, generado automáticamente por el analizador de código de 4D.

En caso contrario, el editor de código muestra [el comentario del bloque en la parte superior del código del método](#).

## Definición del archivo de documentación

4D utiliza una plantilla básica para crear nuevos archivos de documentación. Esta plantilla sugiere las funcionalidades específicas que permiten [mostrar la información en el editor de código](#).

Sin embargo, puede utilizar todas las [etiquetas Markdown soportadas](#).

Los nuevos archivos de documentación se crean con el siguiente contenido por defecto:

```

Method15.md
1 <!-- Type here your summary -->
2 ## Description
3 ~
4 ## Example
5 ~
6 ````4d~
7 Type here your example~
8 ````~

```

Línea	Descripción
"<!-- Type your summary here -->"	Comentario HTML. Se utiliza prioritariamente como descripción del método en los <a href="#">consejos del editor de código</a>
## Description	Título de nivel 2 en Markdown. La primera frase después de esta etiqueta se utiliza como descripción del método en las sugerencias del editor de código si no se utiliza el comentario HTML
## Ejemplo	Título de nivel 2, puede utilizar esta área para mostrar un ejemplo de código
` 4D  Digite su ejemplo aquí`	Se utiliza para dar formato a los ejemplos de código 4D (utiliza la librería highlight.js)

## Markdown soportado

- La etiqueta del título es soportada:

```

# Title 1
## Title 2
### Title 3

```

- Las etiquetas de estilo (cursiva, negrita, tachado) son compatibles:

```

_italic_
**bold**
**_bold/_italic_**
~~strikethrough~~

```

- La etiqueta de bloque de código (` `4d ... ` `) es compatible con el resultado del código 4D:

```
` 4d C_TEXT($txt) $txt:="Hello world!" `
```

- La etiqueta de la tabla es soportada:

Parameter	Type	Description
wpArea	String	Write pro area
toolbar	String	Toolbar name

- La etiqueta del enlace es soportada:

```
// Case 1
The [documentation](https://doc.4d.com) of the command ....

// Case 2
[4D blog] [1]

[1]: https://blog.4d.com
```

- Las etiquetas de imagen son soportadas:

```
![image info](pictures/image.png)

![logo 4D](https://blog.4d.com/wp-content/uploads/2016/09/logoOriginal-1.png "4D blog logo")

![logo 4D blog with link](https://blog.4d.com/wp-content/uploads/2016/09/logoOriginal-1.png "4D blog log")
```



Para más información, consulte la [guía Markdown de GitHub](#).

## Ejemplo

En el archivo `WP_SwitchToolbar.md`, puede escribir:

```
<!-- Este método devuelve un logotipo diferente en función del parámetro de tamaño -->
```

```
GetLogo (size) -> logo

| Parameter | Type      | in/out | Description |
| ----- | ----- | ----- | ----- |
| size     | Longint  | in   | Logo style selector (1 to 5) |
| logo     | Picture   | out  | Selected logo |

## Description
```

Este método devuelve un logotipo diferente en función del parámetro `*size*`.  
`1` = tamaño más pequeño, `5` = tamaño más grande.

## Ejemplo

```
C_PICTURE($logo)
C_LONGINT($size)

//Obtener el logo más grande
$logo:=GetLogo(5)
```

- Vista del Explorador:

HDI\_4DWP\_GetSetFormulas - Explorer

Methods

Project Methods

- 00\_Start
- Compiler\_Arrays
- Compiler\_Methods
- Compiler\_Variables
- Compiler\_Variables\_Inter
- GetEmptyString
- GetFullAddress
- GetLogo
- GetPage
- GetString
- initHDI
- keep
- m\_Quit
- Method15
- zz

Component Methods

Database Methods

Classes

Triggers

Project Form Methods

Table Form Methods

4D Mobile Methods

Parameter	Type	in/out	Description
size	Longint	in	Logo style selector (1 to 5)
logo	Picture	out	Selected logo

## Description

This method returns a logo of a specific size, depending on the value of the `size` parameter value. 1 = smallest size, 5 = largest size.

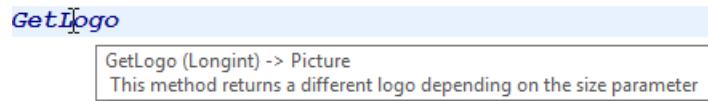
## Example

```
C_PICTURE($logo)
C_LONGINT($size)

//Get the largest logo
$logo:=GetLogo(5)
```

Preview   Documentation

- Vista del Editor de código:



# Acerca del lenguaje 4D

El lenguaje integrado de 4D, que consta de más de 1300 comandos, convierte a 4D en una poderosa herramienta de desarrollo para aplicaciones web, móvil o de escritorio. Puede utilizar el lenguaje 4D para muchas tareas diferentes, desde la realización de cálculos sencillos hasta la creación de complejas interfaces de usuario personalizadas. Por ejemplo, puede:

- Acceder por programación a cualquiera de los editores de gestión de registros (ordenar por, buscar, etc.),
- Cree e imprima informes y etiquetas complejas con la información de la base,
- Comunicarse con otros dispositivos,
- Enviar correos electrónicos,
- Gestionar documentos y páginas web,
- Importación y exportación de datos entre las aplicaciones 4D y otras aplicaciones,
- Incorporar al lenguaje de programación 4D procedimientos escritos en otros lenguajes.

La flexibilidad y el poder del lenguaje de programación 4D lo convierten en la herramienta ideal para todos los niveles de usuarios y desarrolladores para realizar una completa gama de tareas de gestión de la información. Los usuarios principiantes pueden realizar rápidamente los cálculos. Los desarrolladores experimentados pueden utilizar este poderoso lenguaje de programación para añadir sofisticadas funciones y capacidades a sus aplicaciones, como la transferencia de archivos, las comunicaciones y la supervisión. Los desarrolladores con experiencia en programación en otros lenguajes pueden añadir sus propios comandos al lenguaje 4D. Los usuarios experimentados sin experiencia en programación pueden personalizar sus aplicaciones.

## ¿Qué es un lenguaje?

El lenguaje de 4D no es muy diferente del lenguaje hablado que utilizamos a diario. Es una forma de comunicación utilizada para expresar ideas, informar y dar instrucciones. Como un lenguaje hablado, 4D tiene su propio vocabulario, gramática y sintaxis; usted lo utiliza para decirle a 4D cómo manejar su aplicación y sus datos.

No es necesario conocer todo el lenguaje para trabajar eficazmente con 4D. Para hablar, no es necesario conocer todo el idioma inglés; de hecho, se puede tener un vocabulario reducido y seguir siendo bastante elocuente. El lenguaje 4D es muy parecido: sólo se necesita conocer una pequeña parte del lenguaje para ser productivo, y se puede aprender el resto a medida que surja la necesidad.

## ¿Por qué utilizar un lenguaje?

Al principio puede parecer que no es necesario un lenguaje de programación en 4D. En el entorno Diseño, 4D ofrece herramientas flexibles que no requieren programación para realizar una gran variedad de tareas de gestión de datos. Las operaciones fundamentales, como la entrada de datos, las búsquedas, la clasificación y los informes, se realizan con facilidad. De hecho, hay muchas funciones adicionales disponibles, como la validación de datos, las ayudas para la introducción de datos, los gráficos y la generación de etiquetas.

Entonces, ¿por qué necesitamos un lenguaje 4D? Estos son algunos de sus usos:

- Automatizar las tareas repetitivas: por ejemplo, la modificación de datos, la generación de informes complejos y la realización desatendida de largas series de operaciones.
- Controla la interfaz de usuario: puede gestionar las ventanas y los menús, y controlar los formularios y los objetos de la interfaz.
- Realizar una gestión de datos sofisticada: estas tareas incluyen el procesamiento de transacciones, la validación de datos complejos, la gestión multiusuario, los conjuntos y las operaciones de selección temporales.
- Controlar el ordenador: puede controlar las comunicaciones del puerto serie, la gestión de documentos y la gestión de errores.
- Crear aplicaciones: puede crear aplicaciones fáciles de usar y personalizadas que se ejecutan de forma autónoma.
- Añadir funcionalidades al servidor web 4D integrado: construir y actualizar páginas web dinámicas llenas de sus datos.

El lenguaje le permite tener un control total sobre el diseño y el funcionamiento de su aplicación. 4D ofrece poderosos

editores "genéricos", pero el lenguaje le permite personalizar su aplicación al grado que requiera.

## Tomar el control de sus datos

El lenguaje 4D le permite tomar el control total de sus datos de una manera poderosa y elegante. El lenguaje es lo suficientemente fácil para un principiante y lo suficientemente sofisticado para un desarrollador de aplicaciones experimentado. Ofrece transiciones suaves desde las funciones de la base de datos integrada hasta una aplicación completamente personalizada.

Los comandos del lenguaje 4D permiten acceder a los editores estándar de gestión de registros. Puede indicar al comando que busque los datos descritos explícitamente. Puede indicar al comando que busque los datos descritos explícitamente. Por ejemplo, `QUERY( [People]; [People]Last Name="Smith")` encontrará todas las personas de apellido Smith en su base.

El lenguaje 4D es muy poderoso: un solo comando sustituye a menudo cientos o incluso miles de líneas de código escritas en los lenguajes informáticos tradicionales. Sorprendentemente, este poder viene acompañado de la simplicidad: los comandos tienen nombres en inglés. Por ejemplo, para realizar una búsqueda, se utiliza el comando `QUERY`; para añadir un nuevo registro, se utiliza el comando `ADD RECORD`.

El lenguaje está diseñado para que pueda realizar fácilmente casi cualquier tarea. Añadir un registro, ordenar los registros, buscar datos y operaciones similares son definidas con comandos simples y directos. Pero el lenguaje también puede controlar los puertos serie, leer documentos en el disco, controlar el procesamiento de transacciones complejas y mucho más.

El lenguaje 4D realiza incluso las tareas más sofisticadas con relativa sencillez. Realizar estas tareas sin utilizar el lenguaje sería inimaginable para muchos. Incluso con los poderosos comandos del lenguaje, algunas tareas pueden ser complejas y difíciles. Una herramienta por sí misma no hace posible una tarea; la propia tarea puede ser un reto y la herramienta sólo puede facilitar el proceso. Por ejemplo, un procesador de textos hace que escribir un libro sea más rápido y fácil, pero no escribirá el libro por usted. El uso del lenguaje 4D facilitará el proceso de gestión de sus datos y le permitirá abordar tareas complicadas con confianza.

## ¿Es un lenguaje informático "tradicional"?

Si está familiarizado con los lenguajes informáticos tradicionales, esta sección puede ser de interés. Si no es así, es mejor que pase al siguiente párrafo.

El lenguaje 4D no es un lenguaje informático tradicional. Es uno de los lenguajes más innovadores y flexibles que existen actualmente en un ordenador. Está diseñado para adaptarse a usted, y no al revés.

Para utilizar los lenguajes tradicionales, hay que hacer una amplia preparación. De hecho, la planificación es uno de los principales pasos del desarrollo. 4D le permite empezar a utilizar el lenguaje en cualquier momento y en cualquier parte de su proyecto. Puede comenzar añadiendo un método a un formulario, y más tarde añadir algunos métodos más. A medida que su aplicación se vuelve más sofisticada, podría añadir un método proyecto controlado por un menú. Puede utilizar el lenguaje tan poco o tanto como quiera. No es un "todo o nada", como ocurre con muchas otras bases de datos.

Los lenguajes tradicionales obligan a definir y pre-declarar objetos de interfaz en términos sintácticos formales. En 4D, usted simplemente crea un objeto, como un botón, y lo utiliza. 4D gestiona automáticamente el objeto por usted. Por ejemplo, para utilizar un botón, lo dibuja en un formulario y le da un nombre. Cuando el usuario hace clic en el botón, el lenguaje notifica automáticamente sus métodos.

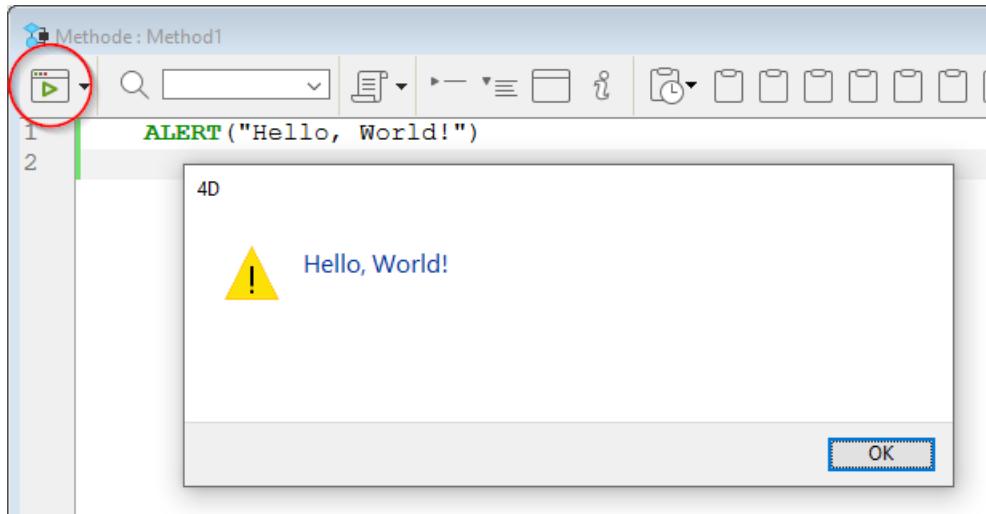
Los lenguajes tradicionales suelen ser rígidos e inflexibles y exigen que los comandos se introduzcan con un estilo muy formal y restrictivo. El lenguaje 4D rompe con la tradición, y los beneficios son para usted.

# Un recorrido rápido

Utilizando el lenguaje 4D, la impresión del tradicional mensaje "Hello, world!" en pantalla puede hacerse de varias maneras. Lo más sencillo es probablemente escribir la siguiente línea única en un método de proyecto:

```
ALERT("Hello, World!")
```

Este código mostrará una caja de diálogo de alerta estándar de la plataforma con el mensaje "Hello, World!", que contiene un botón de OK. Para ejecutar el código, basta con hacer clic en el botón de ejecución en el editor de métodos:



O bien, podría adjuntar este código a un botón de formulario y ejecutarlo, en cuyo caso al hacer clic en el botón se mostraría la caja de diálogo de alerta. En todo caso, ¡acaba de ejecutar su primera línea de código 4D!

## Asignar los valores

Los datos pueden introducirse y copiarse en variables, campos, elementos de arrays... Poner datos en una variable se llama asignar los datos a la variable y se hace con el operador de asignación (:=). El operador de asignación también se utiliza para asignar datos a campos o elementos de arrays.

```
$MyNumber:=3 //asigna 3 a la variable MyNumber  
[Products]Size:=$MyNumber //asigna la variable MyNumber al campo [Products]Size  
arrDays{2}:="Tuesday" //asigna la cadena "Tuesday" al segundo elemento de arrDays  
MyVar:=Length("Acme") //asigna el resultado de la función (4) a MyVar  
$myDate:=!2018/01/21! //asigna una fecha literal  
$myHour:?=08:12:55? //asigna una hora literal //asigna una fecha literal  
$myHour:?=08:12:55? //asigna una hora literal //asigna una fecha literal  
$myHour:?=08:12:55? //asigna una hora literal //asigna una fecha literal  
$myHour:?=08:12:55? //asigna una hora literal
```

Debe distinguir el operador de asignación := de los demás operadores. En lugar de combinar expresiones en una nueva expresión, el operador de asignación copia el valor de la expresión a la derecha del operador de asignación en la variable o campo a la izquierda del operador.

Importante: no confunda el operador de asignación (:=) con el signo igual (=). Se ha elegido deliberadamente un operador de asignación diferente (y no =) para evitar los problemas y la confusión que suelen producirse con == o === en otros lenguajes de programación. Estos errores son a menudo difíciles de reconocer por el compilador y conducen a una solución de problemas que requiere mucho tiempo.

# Variables

El lenguaje 4D es estricto con los tipos de datos, aunque se permite cierta flexibilidad en muchos casos. Por ejemplo, para crear una variable de tipo fecha, puede escribir: Se crea una variable digitada utilizando la palabra clave `var`.

```
var MyDate : Date
```

La palabra clave `var` permite declarar variables objeto de un tipo de clase definido, por ejemplo:

```
var myPerson : cs.Person  
//de la clase de usuario Person
```

Aunque no se suele recomendar, se pueden declarar variables simplemente utilizándolas; no es necesario definirlas formalmente. Por ejemplo, si desea una variable que contenga la fecha actual más 30 días, puede escribir:

```
MyOtherDate:=Current date+30
```

La línea de código dice "MyOtherDate obtiene la fecha actual más 30 días" Esta línea declara la variable, la asigna con el tipo de fecha (temporal) y un contenido. Esta línea declara la variable, la asigna con el tipo de fecha (temporal) y un contenido. Una variable declarada por asignación se interpreta como sin tipo, es decir, puede ser asignada con otros tipos en otras líneas y entonces cambia el tipo dinámicamente. Una variable digitada con `var` no puede cambiar de tipo. Sin embargo, en [modo compilado](#), el tipo nunca puede cambiarse, independientemente de cómo se haya declarado la variable.

# Comandos

Los comandos 4D son métodos integrados para realizar una acción. Todos los comandos 4D, como `CREATE RECORD`, o `ALERT`, se describen en el manual *Lenguaje de 4D*, agrupados por temas. Los comandos se utilizan a menudo con parámetros, que se pasan entre corchetes () y separados por punto y coma (;). Ejemplo:

```
COPY DOCUMENT("folder1\\name1";"folder2\\" ; "new")
```

Algunos comandos se adjuntan a colecciones u objetos, en cuyo caso son métodos temporales que se utilizan con la notación de puntos. Por ejemplo:

```
$c:=New collection(1;2;3;4;5)  
$nc:=$c.slice(0;3) // $nc=[1,2,3]  
  
$lastEmployee:=$employee.last()
```

Puede utilizar los plug-ins o los componentes 4D que añaden nuevos comandos a su entorno de desarrollo 4D.

Hay muchos plug-ins propuestos por la comunidad de usuarios de 4D o por desarrolladores terceros. Por ejemplo, utilizando el [4d-plugin-pdf-pages](#) en macOS:

```
PDF REMOVE PAGE(path;page)
```

4D SVG es un ejemplo de componente utilitario que aumenta las capacidades de su aplicación:

```
//hacer un dibujo  
svgRef:=SVG_New  
objectRef:=SVG_New_arc(svgRef;100;100;90;90;180)
```

4D SVG está incluido en 4D.

## Constantes

4D ofrece un conjunto extensivo de constantes predefinidas, cuyos valores son accesibles por nombre. Permiten escribir un código más legible. Por ejemplo, `Read Mode` es una constante (valor 2).

```
vRef:=Open document("PassFile";"TEXT";Read Mode) // abrir el documento en modo de sólo lectura
```

Las constantes predefinidas aparecen subrayadas por defecto en el editor de métodos 4D.

## Métodos

4D ofrece un gran número de métodos (o comandos) integrados, pero también le permite crear sus propios métodos de proyecto. Los métodos de proyecto son métodos definidos por el usuario que contienen comandos, operadores y otras partes del lenguaje. Los métodos proyecto son métodos genéricos, pero hay otros tipos de métodos: métodos objeto, métodos formulario, métodos tabla (Triggers) y métodos base.

Un método se compone de varias líneas de instrucciones, cada una de las cuales consta de una línea en el método. Una línea de instrucción realiza una acción, y puede ser simple o compleja.

Por ejemplo, la siguiente línea es una sentencia que mostrará una caja de diálogo de confirmación:

```
CONFIRM("¿Realmente quiere cerrar esta cuenta?"; "Sí"; "No")
```

Un método también contiene pruebas y bucles que controlan el flujo de ejecución. Los métodos 4D soportan las estructuras `If...Else...End if` y `Case of... Else...End case`, así como los bucles: `While...End while`, `Repeat...Until`, `For...End for`, y `For each... End for each`:

El siguiente ejemplo recorre todos los caracteres del texto `vtSomeText`:

```
For($vlChar;1;Length(vtSomeText))
    //Hacer algo con el carácter si es un TAB
    If(Character code(vtSomeText[[${vlChar}]))=Tab)
        //...
    End if
End for
```

Un método proyecto puede llamar a otro método proyecto con o sin parámetros (argumentos). Los parámetros se pasan al método entre paréntesis, a continuación del nombre del método. Cada parámetro está separado del siguiente por un punto y coma (;). Los parámetros están disponibles dentro del método llamado como variables locales numeradas secuencialmente: \$1, \$2,..., \$n. Un método puede devolver un único valor en el parámetro \$0. Cuando se llama a un método, sólo hay que escribir su nombre:

```
$myText:="hello"
$myText:=Do_Something($myText) //Llama al método Do_Something
ALERT($myText) //"HELLO"

//Este es el código del método Do_Something
$0:=Uppercase($1)
```

## Tipos de datos

En el lenguaje, los distintos tipos de datos que se pueden manejar se denominan tipos de datos. Existen tipos de datos

básicos (cadena, numérico, fecha, hora, booleano, imagen, punteros, arrays), y también tipos de datos compuestos (BLOBs, objetos, colecciones).

Tenga en cuenta que los datos de tipo cadena y numérico pueden asociarse a más de un tipo de campo. Cuando se introducen datos en un campo, el lenguaje convierte automáticamente los datos en el tipo correcto para el campo. Por ejemplo, si se utiliza un campo entero, sus datos se tratan automáticamente como numéricos. En otras palabras, no tiene que preocuparse por mezclar tipos de campos similares al utilizar el lenguaje; éste los gestionará por usted.

Sin embargo, al utilizar el lenguaje es importante no mezclar los diferentes tipos de datos. Del mismo modo que no tiene sentido almacenar "ABC" en un campo de fecha, tampoco tiene sentido poner "ABC" en una variable utilizada para fechas. En la mayoría de los casos, 4D es muy tolerante y tratará de dar sentido a lo que está haciendo. Por ejemplo, si añade un número a una fecha, 4D asumirá que quiere añadir ese número de días a la fecha, pero si intenta añadir una cadena a una fecha, 4D le dirá que la operación no puede funcionar.

Hay casos en los que es necesario almacenar datos como un tipo y utilizarlos como otro. El lenguaje contiene un conjunto completo de comandos que permiten convertir de un tipo de datos a otro. Por ejemplo, es posible que necesite crear un número de pieza que empiece por un número y termine con caracteres como "abc". En este caso, podría escribir:

```
[Products]Part Number:=String(Number)+"abc"
```

Si *Number* es 17, then *[Products]Part Number* obtendrá el valor "17abc".

Los tipos de datos están completamente definidos en la sección [Tipos de datos](#).

## Objetos y colecciones

Puedes manejar objetos y colecciones del lenguaje 4D utilizando la notación objeto para obtener o definir sus valores. Por ejemplo:

```
employee.name:="Smith"
```

También puede utilizar una cadena entre corchetes, por ejemplo:

```
$vName:=employee["name"]
```

Como el valor de una propiedad de objeto puede ser un objeto o una colección, la notación objeto acepta una secuencia de símbolos para acceder a subpropiedades, por ejemplo:

```
$vAge:=employee.children[2].age
```

Tenga en cuenta que si el valor de la propiedad del objeto es un objeto que encapsula un método (una fórmula), debe añadir paréntesis () al nombre de la propiedad para ejecutar el método:

```
$f:=New object
$f.message:=New formula(ALERT("Hello world!"))
$f.message() //displays "Hello world!"
$f.message() //displays "Hello world!"
$f.message() //displays "Hello world!"
$f.message() //displays "Hello world!"
```

Para acceder a un elemento de la colección, debe pasar el número del elemento entre corchetes:

```
C_COLLECTION(myColl)
myColl:=New collection("A";"B";1;2;Current time)
myColl[3] //acceso al 4º elemento de la colección
```

## Clases

El lenguaje 4D soporta las clases de objetos. Añade un archivo `myClass.4dm` en la carpeta Project/Sources/Classes de un proyecto para crear una clase llamada "myClass".

Para instanciar un objeto de la clase en un método, llame la clase usuario desde el `class store` (`cs`) y utilice la función miembro `new()`. Se pueden pasar parámetros.

```
// en un método 4D
$o:=cs.myClass.new()
```

En el método de clase `myClass`, utilice la instrucción `Function <methodName>` para definir el método miembro de clase `methodName`. Un método miembro de clase puede recibir y devolver parámetros como cualquier método, y utilizar `This` como instancia del objeto.

```
//en el archivo myClass.4dm
Function hello
  C_TEXT($0)
  $0:="Hello "+This.who
```

Para ejecutar un método miembro de clase, basta con utilizar el operador `()` en el método miembro de la instancia del objeto.

```
$o:=cs.myClass.new()
$o.who:="World"
$message:=$o.myClass.hello()
//$message: "Hello World"
```

Opcionalmente, utilice la palabra clave `Class constructor` para declarar las propiedades del objeto.

```
/en el archivo Rectangle.4dm
Class constructor
C_LONGINT($1;$2)
This.height:=$1
This.width:=$2
This.name:="Rectangle"
```

Una clase puede extender otra clase utilizando `Class extends <ClassName>`. Las superclasses se pueden llamar con el comando `Super`. Por ejemplo:

```
//en el archivo Square.4dm
Class extends rectangle

Class constructor
C_LONGINT($1)

//Llama al constructor de la clase padre con las dimensiones
// suministradas para el ancho y el alto del Rectángulo
Super($1;$1)

This.name:="Square"
```

## Operadores

Cuando se utiliza el lenguaje, es raro que se quiera simplemente un dato. Es más probable que quiera hacer algo con esos datos. Estos cálculos se realizan con operadores. Los operadores, en general, toman dos datos y realizan una operación sobre ellos que da como resultado un nuevo dato. Usted ya conoce a la mayoría de los operadores. Por ejemplo, `1 + 2` utiliza el operador de adición (o signo más) para sumar dos números, y el resultado es 3. Esta tabla muestra algunos operadores numéricos comunes:

Operador	Operación	Ejemplo
+	Adición	<code>1 + 2 = 3</code>
-	Resta	<code>3 - 2 = 1</code>
*	Multiplicación	<code>2 * 3 = 6</code>
/	División	<code>6 / 2 = 3</code>

Los operadores numéricos son sólo un tipo de operador disponible. 4D soporta múltiples tipos de datos, como números, texto, fechas e imágenes, por lo que existen operadores que realizan operaciones con estos diferentes tipos de datos.

Los mismos símbolos se utilizan a menudo para diferentes operaciones, dependiendo del tipo de datos. Por ejemplo, el signo más (+) realiza diferentes operaciones con diferentes datos:

Tipos de datos	Operación	Ejemplo
Número	Adición	<code>1 + 2</code> suma los números y da como resultado 3
Cadena	Concatenación	<code>"Hola" + "a todos"</code> concatena (une) las cadenas y da como resultado "Hola a todos"
Fecha y Número	Adición de fecha	<code>!1989-01-01! + 20</code> añade 20 días a la fecha del 1 de enero de 1989 y da como resultado la fecha del 21 de enero de 1989

## Expresiones

En pocas palabras, las expresiones devuelven un valor. De hecho, al utilizar el lenguaje 4D, se utilizan expresiones todo el tiempo y se tiende a pensar en ellas sólo en términos del valor que representan. Las expresiones también se llaman fórmulas.

Las expresiones se componen de casi todas las demás partes del lenguaje: comandos, operadores, variables, campos, propiedades de objetos y elementos de colección. Se utilizan expresiones para escribir líneas de código, que a su vez se utilizan para construir métodos. El lenguaje utiliza expresiones siempre que necesita un dato.

Las expresiones son rara vez "autónomas." Hay varios lugares en 4D donde una expresión puede ser utilizada por sí misma. Incluye:

- Editor de fórmulas (apply formula, query with formula, order by formula)
- El comando EXECUTE FORMULA

- La lista de propiedades, donde se puede utilizar una expresión como fuente de datos para la mayoría de los widgets
- Depurador donde se puede comprobar el valor de las expresiones
- En el editor de informes rápidos como fórmula para una columna

## Tese de expresiones

Se hace referencia a una expresión por el tipo de datos que devuelve. Hay varios tipos de expresiones. En la siguiente tabla se dan ejemplos de cada tipo de expresión.

Expresión	Tipo	Descripción
"Hello"	Cadena	La palabra Hola es una constante cadena, indicada por las comillas dobles.
"Hello " + "there"	Cadena	Dos cadenas, "Hola" y "a todos", se suman (concatenan) con el operador de concatenación de cadenas (+). Se devuelve la cadena "Hola".
"Sr. " + [People]Name	Cadena	Se concatenan dos cadenas: la cadena "Sr." y el valor actual del campo Nombre de la tabla Personas. Si el campo contiene "Smith", la expresión devuelve "Mr. Smith".
Uppercase("smith")	Cadena	Esta expresión utiliza <code>Uppercase</code> , un comando del lenguaje, para convertir la cadena "smith" a mayúsculas. Devuelve "SMITH".
4	Número	Se trata de una constante numérica, 4.
4 * 2	Número	Dos números, 4 y 2, se multiplican utilizando el operador de multiplicación (*). El resultado es el número 8.
myButton	Número	Es una variable asociada a un botón. Devuelve el valor actual del botón: 1 si se ha hecho clic, 0 si no.
!1997-01-25!	Fecha	Esta es una constante fecha para la fecha 1/25/97 (25 de enero de 1997).
Current date+ 30	Fecha	Esta es una expresión de tipo Fecha que utiliza el comando <code>Current date</code> para obtener la fecha de hoy. Añade 30 días a la fecha de hoy y devuelve la nueva fecha.
?8:05:30?	Hora	Es una constante hora que representa 8 horas, 5 minutos y 30 segundos.
?2:03:04? + ?1:02:03? ? 2:03:04? + ?1:02:03? ?2:03:04? + ?1:02:03? + ?1:02:03?	Hora	Esta expresión suma dos horas y devuelve la hora 3:05:07.
True	Booleano	Este comando devuelve el valor booleano TRUE.
10 # 20	Booleano	Se trata de una comparación lógica entre dos números. El símbolo número (#) significa "es diferente de". Como 10 "es diferente de" 20, la expresión devuelve TRUE.
"ABC" = "XYZ"	Booleano	Se trata de una comparación lógica entre dos cadenas. Son diferentes, por lo que la expresión devuelve FALSE.
My Picture + 50	Imagen	Esta expresión toma la imagen en My Picture, la mueve 50 píxeles a la derecha y devuelve la imagen resultante.
->[People]Name	Puntero	Esta expresión devuelve un puntero al campo llamado [People]Name.
Table (1)	Puntero	Este es un comando que devuelve un puntero a la primera tabla.
JSON Parse (MyString)	Objeto	Este es un comando que devuelve MyString como un objeto (si el formato es el adecuado)
JSON Parse (MyJSONArray)	Collection	Este es un comando que devuelve MyJSONArray en forma de

Expresión	Tipo	Descripción
Form.pageNumber	Propiedad objeto	Una propiedad objeto es una expresión que puede ser de todo tipo soportado
Col[5]	Elementos de colección	Un elemento de colección es una expresión que puede ser de todo tipo soportado
\$entitySel[0]	Entity	Un elemento de una selección de entidades ORDA es una expresión de tipo entidad. Este tipo de expresión es no assignable

## Expresiones asignables y no asignables

Una expresión puede ser simplemente una constante literal, como el número 4 o la cadena "Hello", o una variable como \$myButton . También puede utilizar los operadores. Por ejemplo, 4 + 2 es una expresión que utiliza el operador de adición para sumar dos números y devolver el resultado 6. En todos los casos, estas expresiones son no asignables, lo que significa que no se les puede asignar un valor. En 4D, las expresiones pueden ser asignables. Una expresión es assignable cuando puede utilizarse a la izquierda del operador de asignación. Por ejemplo:

```
//La variable $myVar es assignable, puede escribir:  
$myVar:="Hola" //asignar "Hola" a myVar  
//Form.pageNumber es assignable, puede escribir: Form.pageNumber:=10 //asignar 10 a Form.pageNumber  
//Form.pageTotal-Form.pageNumber no es assignable: Form.pageTotal- Form.pageNumber:=10 //error, no asigna
```

En general, las expresiones que utilizan un operador no son asignables. Por ejemplo, [Person]FirstName+" "+ [Person]LastName no es assignable.

## Punteros

El lenguaje 4D ofrece una implementación avanzada de punteros, que permite escribir código poderoso y modular. Puede utilizar punteros para referenciar tablas, campos, variables, arrays y elementos de arrays.

Un puntero a un elemento se crea añadiendo un símbolo "->" antes del nombre del elemento, y se puede desreferenciar añadiendo el símbolo "->" después del nombre del puntero.

```
MyVar:="Hello"  
MyPointer:=->MyVar  
ALERT(MyPointer->)
```

## Comentarios

Los comentarios son líneas de instrucciones inactivas. Estas líneas no son interpretadas por el programa 4D y no se ejecutan cuando el código se llama.

Hay dos maneras de crear comentarios:

- // para crear una línea de comentario
- /\*...\*/ para los bloques de comentarios en línea o multilínea.

Ambos estilos de comentarios pueden utilizarse simultáneamente.

### Línea de comentario (//)

Inserte // al principio de una línea o después de una instrucción para añadir una línea de comentario. Ejemplo:

```
//Este es un comentario
For($vCounter;1;100) //Inicio del bucle
    //comment
    //comment
    //comment
End for
```

## Comentarios en línea o multilínea /\* \*/

Rodea el contenido con los caracteres `/* ... */` para crear comentarios en línea o bloques de comentarios multilínea. Tanto los bloques de comentarios en línea como los multilínea comienzan con `/*` y terminan con `*/`.

- Las líneas de comentarios en línea se pueden insertar en cualquier parte del código. Ejemplo:

```
For /* línea de comentario */ ($vCounter;1;100)
    ...
End for
```

- Los bloques de comentarios multilíneas permiten comentar un número ilimitado de líneas. Los bloques de comentarios pueden anidarse (útil desde que el editor de código 4D soporta los bloques contraídos). Ejemplo:

```
For ($vCounter;1;100)
/*
comentarios
/*
    otros comentarios
*/
*/
...
End for
```

# Operadores

An operator is a symbol or a group of symbols that you use to check, modify, or combine values. Usted ya conoce a la mayoría de los operadores. For example, `1 + 2` uses the addition (or plus sign) operator to add two numbers together, and the result is 3. Comparison operators, like `=` or `>`, let you compare two or more values.

The 4D language supports the operators you may already know from other languages like C or JavaScript. However, the assignment operator is `:=` to prevent it from being mistakenly used when the equal to operator (`=`) is intended. [Basic operators](#) such as arithmetic operators (`+`, `-`, `*`, `/`, `%`...) and comparison operators (`=`, `>`, `>=`...) can be used with numbers, but also with boolean, text, date, time, pointer, or picture data types. Like JavaScript, the 4D language supports the concept of [truthy and falsy values](#), which is used in [short-circuit operators](#).

## Terminología

The 4D language supports binary and ternary operators:

- los operadores binarios operan en dos objetivos (como `2 + 3`) y aparecen entre sus dos objetivos.
- los operadores ternarios operan en tres objetivos. Like C, 4D has only one ternary operator, the [ternary conditional operator](#) (`a ? b : c`).

Los valores que los operadores afectan son los operandos. En la expresión `1 + 2`, el símbolo `+` es un operador binario y sus dos operandos son los valores 1 y 2.

## Asignación

The assignment operator (`a:=b`) initializes or updates the value of `a` with the value of `b`:

```
$myNumber:=3 //assigns 3 to MyNumber variable
$myDate:=!2018/01/21! $myNumber:=3 //assigns 3 to MyNumber variable
$myDate:=!2018/01/21! //assigns a date literal
$myLength:=Length("Acme") //assigns the result of the command (4) to $myLength
$col:=New collection // $col is initialized with an empty collection $myNumber:=3 //assigns 3 to MyNumber
$myDate:=!2018/01/21! //assigns a date literal
$myLength:=Length("Acme") //assigns the result of the command (4) to $myLength
$col:=New collection // $col is initialized with an empty collection $myNumber:=3 //assigns 3 to MyNumber
$myDate:=!2018/01/21! //assigns a date literal
$myLength:=Length("Acme") //assigns the result of the command (4) to $myLength
$col:=New collection // $col is initialized with an empty collection
```

Do NOT confuse the assignment operator `:=` with the equality comparison operator `=`. A different assignment operator (and not `=`) was deliberately chosen to avoid issues and confusion which often occur with `==` or `====` in other programming languages. Estos errores son a menudo difíciles de reconocer por el compilador y conducen a una solución de problemas que requiere mucho tiempo.

## Operadores básicos

Operator results depend on the data types they are applied to. 4D supports different operators on scalar data types. They are described with the data types, in the following sections:

- [Logical operators](#) (on boolean expressions)

- [Operadores de fechas](#)
- [Operadores de horas](#)
- [Operadores numéricos](#)
- [Bitwise operators](#) (on long integer expressions)
- [Operadores de imágenes](#)
- [Operadores en punteros](#)
- [Operadores de cadenas](#)

## Operadores de asignación compuestos

4D provides compound assignment operators that combine assignment with another operation. One example is the addition assignment operator ( `+=` ):

```
$a:=1  
$a+=2 // $a=3
```

The following compound assignment operators are supported:

Operador	Sintaxis	Asigna	Ejemplo
Adición	Text += Text	Texto	<code>\$t+=" World" // \$t:=\$t+" World"</code>
	Number += Number	Número	<code>\$n+=5 // \$n:=\$n+5</code>
	Date += Number	Fecha	<code>\$d+=5 // \$d:=\$d+5</code>
	Time += Time	Hora	<code>\$t1+= \$t2 // \$t1:=\$t1+\$t2</code>
	Time += Number	Número	<code>\$t1+=5 // \$t1:=\$t1+5</code>
	Picture += Picture	Imagen	<code>\$p1+= \$p2 // \$p1:=\$p1+\$p2 (add \$p2 to the right of \$p1)</code>
Resta	Number -= Number	Número	<code>\$n-=5 // \$n:=\$n-5</code>
	Date -= Number	Fecha	<code>\$d-=5 // \$d:=\$d-5</code>
	Time -= Time	Hora	<code>\$t1-= \$t2 // \$t1:=\$t1-\$t2</code>
	Time -= Number	Número	<code>\$t1-=5 // \$t1:=\$t1-5</code>
	Picture -= Number	Imagen	<code>\$p1-=5 // \$p1:=\$p1-5 (move \$p1 horizontally 5 pixels to the left)</code>
	División	Número	<code>\$n/=5 // \$n:=\$n/5</code>
Multiplicación	Number /= Number	Número	<code>\$t1/= \$t2 // \$t1:=\$t1/\$t2</code>
	Time /= Time	Hora	<code>\$t1/=5 // \$t1:=\$t1/5</code>
	Picture /= Picture	Imagen	<code>\$p1/= \$p2 // \$p1:=\$p1/\$p2 (add \$p2 to the bottom of \$p1)</code>
	Picture /= Number	Imagen	<code>\$p1/=5 // \$p1:=\$p1/5 (desplazar verticalmente \$p1 de 5 píxeles)</code>
	Text *= Number	Texto	<code>\$t*="abc" // \$t:=\$t*"abc"</code>
	Number *= Number	Número	<code>\$n*=5 // \$n:=\$n*5</code>
	Time *= Time	Hora	<code>\$t1*=\$t2 // \$t1:=\$t1*\$t2</code>
	Time *= Number	Número	<code>\$t1*=5 // \$t1:=\$t1*5</code>
	Picture *= Number	Imagen	<code>\$p1*=5 // \$p1:=\$p1*5 (redimensionar \$p1 de 5)</code>

These operators apply on any [assignable expressions](#) (except pictures as object properties or collection elements).

The operation "source operator value" is not strictly equivalent to "source := source operator value" because the expression designating the source (variable, field, object property, collection element) is only evaluated once. For example, in such expression as `getPointer()=>+=1` the `getPointer` method is called only once.

Character indexing in text and byte indexing in blob do not support these operators.

## Ejemplos

```
// Addition
$x:=2
$x+=5 // $x=7
```

```

$t:="Hello"
$t+=" World" // $t="Hello World"

$d:!=2000-11-10!
$d+=10 // $d=2000-11-20!

// Subtraction
$x1:=10
$x1-=5 // $x1=5

$d1:!=2000-11-10!
$d1-=10 // $d1=2000-10-31!

// Division
$x3:=10
$x3/=2 // $x3=5

// Multiplication
$x2:=10
$x2*=5 // $x2=10

$t2:="Hello"
$t2*=2 // $t2="HelloHello"
$d+=10 // $d=2000-11-20!

// Subtraction
$x1:=10
$x1-=5 // $x1=5

$d1:!=2000-11-10!
$d1-=10 // $d1=2000-10-31!

// Division
$x3:=10
$x3/=2 // $x3=5

// Multiplication
$x2:=10
$x2*=5 // $x2=10

$t2:="Hello"
$t2*=2 // $t2="HelloHello"
$d+=10 // $d=2000-11-20!

// Resta
$x1:=10
$x1-=5 // $x1=5

$d1:!=2000-11-10!
$d1-=10 // $d1=2000-10-31!

// Division
$x3:=10
$x3/=2 // $x3=5

// Multiplication
$x2:=10
$x2*=5 // $x2=10

$t2:="Hello"
$t2*=2 // $t2="HelloHello"
$d+=10 // $d=2000-11-20!

// Resta
$x1:=10
$x1-=5 // $x1=5

```

✓ // -- ✓ // -- ✓

```
$d1:=!2000-11-10!
$d1-=10 // $d1=!2000-10-31!

// Division
$x3:=10
$x3/=2 // $x3=5

// Multiplication
$x2:=10
$x2*=5 // $x2=10

$t2:="Hello"
$t2*=2 // $t2="HelloHello"
$d+=10 // $d=!2000-11-20!

// Resta
$x1:=10
$x1-=5 // $x1=5

$d1:=!2000-11-10!
$d1-=10 // $d1=!2000-10-31!

// Division
$x3:=10
$x3/=2 // $x3=5

// Multiplication
$x2:=10
$x2*=5 // $x2=10

$t2:="Hello"
$t2*=2 // $t2="HelloHello"
```

## Short-circuit operators

The `&&` and `||` operators are short circuit operators. A short circuit operator is one that doesn't necessarily evaluate all of its operands.

The difference with the single [& and | boolean operators](#) is that the short-circuit operators `&&` and `||` don't return a boolean value. They evaluate expressions as [truthy or falsy](#), then return one of the expressions.

### Short-circuit AND operator (`&&`)

La regla es la siguiente:

Dado `Expr1 && Expr2` :

The short-circuit AND operator evaluates operands from left to right, returning immediately with the value of the first falsy operand it encounters; if all values are [truthy](#), the value of the last operand is returned.

The following table summarizes the different cases for the `&&` operator:

Expr1	Expr2	Valor devuelto
truthy	truthy	Expr2
truthy	falsy	Expr2
falsy	truthy	Expr1
falsy	falsy	Expr1

## Ejemplo 1

```
var $v : Variant

$v:= "Hello" && "World" //"World"
$v:=False && 0 // False
$v:=0 && False // False
$v:=5 && !00-00-00! // 00/00/00
$v := 5 && 10 && "hello" //"hello" // 00/00/00
$v := 5 && 10 && "hello" //"hello" // 00/00/00
$v := 5 && 10 && "hello" //"hello" // 00/00/00
$v := 5 && 10 && "hello" //"hello" // 00/00/00
$v := 5 && 10 && "hello" //"hello"
```

## Ejemplo 2

Say you have an online store, and some products have a tax rate applied, and others don't.

To calculate the tax, you multiply the price by the tax rate, which may not have been specified.

Así que puede escribir esto:

```
var $tax : Variant

$tax:=$item.taxRate && ($item.price*$item.taxRate)
```

\$tax will be NULL if taxRate is NULL (or undefined), otherwise it will store the result of the calculation.

## Ejemplo 3

Short-circuit operators are useful in tests such as:

```
If(($myObject#Null) && ($myObject.value>10))
    //code
End if
```

If \$myObject is Null, the second argument is not executed, thus no error is thrown.

## Short-circuit OR operator (||)

The || operator returns the value of one of the specified operands. The expression is evaluated left to right and tested for possible "short-circuit" evaluation using the following rule:

Given Expr1 || Expr2 :

If Expr1 is **truthy**, Expr2 is not evaluated and the calculation returns Expr1.

If Expr1 is **falsy**, the calculation returns Expr2.

The following table summarizes the different cases and the value returned for the || operator:

Expr1	Expr2	Valor devuelto
truthy	truthy	Expr1
truthy	falsy	Expr1
falsy	truthy	Expr2
falsy	falsy	Expr2

## Ejemplo 1

Supongamos que tiene una tabla llamada Employee. Some employees have entered a phone number, and others haven't. This means that `$emp.phone` could be NULL, and you cannot assign NULL to a Text variable. Pero puede escribir lo siguiente:

```
var $phone : Text  
  
$phone:=$emp.phone || "n/a"
```

In which case `$phone` will store either a phone number or the "n/a" string.

## Ejemplo 2

Given a table called Person with a `name` field, as well as a `maiden name` field for married women.

The following example checks if there is a maiden name and stores it in a variable, otherwise it simply stores the person's name:

```
var $name: Text  
  
$name:=$person.maidenName || $person.name
```

## Prioridad

The `&&` and `||` operators have the same precedence as the logical operators `&` and `|`, and are evaluated left to right.

This means that `a || b && c` is evaluated as `(a || b) && c`.

## Operador ternario

The ternary conditional operator allows you to write one-line conditional expressions. For example, it can replace a full sequence of [If...Else](#) statements.

Se necesitan tres operandos en el siguiente orden:

- una condición seguida de un signo de interrogación (?)
- an expression to execute if the condition is `truthy`, followed by a colon (:)
- an expression to execute if the condition is `falsy`

## Sintaxis

La sintaxis es la siguiente:

```
condition ? condition ? exprIfTruthy : exprIfFalsy condition ? exprIfTruthy : exprIfFalsy condition ?  
exprIfTruthy : exprIfFalsy
```

Since the [token syntax](#) uses colons, we recommend inserting a space after the colon `:` or enclosing tokens using parentheses to avoid any conflicts.

## Ejemplos

Un ejemplo sencillo

```

var $age : Integer
var $beverage : Text

$age:=26
$beverage:=($age>=21) ? "Beer" : "Juice"

ALERT($beverage) // "Beer" "Beer" : "Juice"

ALERT($beverage) // "Beer"

```

## Gestión de los datos de una tabla

This example stores a person's full name in a variable, and handles the case when no first name or last name has been specified:

```

var $fullname : Text

// If one of the names is missing, store the one that exists, otherwise store an empty string
$fullname:=($person.firstname && $person.lastname) ? ($person.firstname+" "+$person.lastname) : ($person

```

## Truthy y falsy

As well as a type, each value also has an inherent Boolean value, generally known as either `truthy` or `falsy`.

truthy and falsy values are only evaluated by [short-circuit](#) and [ternary](#) operators.

The following values are falsy:

- `false`
- `Null`
- `indefinido`
- `Null object`
- `Null collection`
- `Null pointer`
- `Null picture`
- `Null date !00-00-00!`
- `"" - Cadenas vacías`
- `[] - Colecciones vacías`
- `{ } - Objetos vacíos`

All other values are considered truthy, including:

- `0 - cero numérico (Entero u otro)`

In 4D, truthy and falsy evaluation reflects the usability of a value, which means that a truthy value exists and can be processed by the code without generating errors or unexpected results. The rationale behind this is to provide a convenient way to handle `undefined` and `null` values in objects and collections, so that a reduced number of [If...Else](#) statements are necessary to avoid runtime errors.

For example, when you use a [short-circuit OR operator](#):

```
$value:=$object.value || $defaultValue
```

... you get the default value whenever `$object` does not contain the `value` property OR when it is `null`. So this operator checks the existence or usability of the value instead of a specific value. Note that because the numerical value 0 exists and is usable, it is not treated specially, thus it is truthy.

Regarding values representing collections, objects, or strings, "empty" values are considered falsy. It is handy when you want to assign a default value whenever an empty one is encountered.

```
$phone:=$emp.phone || "n/a"
```

# Tipos de datos

En 4D, los datos se manejan según su tipo en dos lugares: los campos de la base y el lenguaje 4D.

Aunque suelen ser equivalentes, algunos tipos de datos disponibles en la base no están disponibles directamente en el lenguaje y se convierten automáticamente. Por el contrario, algunos tipos de datos sólo pueden manejarse a través del lenguaje. La siguiente tabla lista todos los tipos de datos disponibles y cómo se soportan/declaran:

Tipos de datos	Soporte para la base (1)	Soporte para el lenguaje	declaración <code>var</code>	declaración <code>C_</code> o <code>ARRAY</code>
Alfanumérico	Sí	Convertido en texto	-	-
Texto	Sí	Sí	Texto	<code>C_TEXT</code> , <code>ARRAY TEXT</code>
Fecha	Sí	Sí	Fecha	<code>C_DATE</code> , <code>ARRAY DATE</code>
Hora	Sí	Sí	Hora	<code>C_TIME</code> , <code>ARRAY TIME</code>
Booleano	Sí	Sí	Booleano	<code>C_BOOLEAN</code> , <code>ARRAY BOOLEAN</code>
Integer	Sí	Convertido en entero largo	Integer	<code>ARRAY INTEGER</code>
Entero largo	Sí	Sí	Integer	<code>C_LONGINT</code> , <code>ARRAY LONGINT</code>
Entero largo 64 bits	Sí (SQL)	Convertido en real	-	-
Real	Sí	Sí	Real	<code>C_REAL</code> , <code>ARRAY REAL</code>
Indefinido	-	Sí	-	-
Null	-	Sí	-	-
Puntero	-	Sí	Puntero	<code>C_POINTER</code> , <code>ARRAY POINTER</code>
Imagen	Sí	Sí	Imagen	<code>C_PICTURE</code> , <code>ARRAY PICTURE</code>
BLOB	Sí	Sí	<code>Blob</code> , <code>4D.Blob</code>	<code>C_BLOB</code> , <code>ARRAY BLOB</code>
Objeto	Sí	Sí	Objeto	<code>C_OBJECT</code> , <code>ARRAY OBJECT</code>
Collection	-	Sí	Collection	<code>C_COLLECTION</code>
Variant(2)	-	Sí	Variant	<code>C_VARIANT</code>

(1) Tenga en cuenta que ORDA maneja los campos de la base a través de objetos (entidades) y por lo tanto, sólo soporta los tipos de datos disponibles para estos objetos. Para más información, consulte la descripción del tipo de datos [Objeto](#).

(2) La variante no es en realidad un tipo de *datos* sino un tipo de *variable* que puede contener un valor de cualquier otro tipo de datos.

## Valores por defecto

Cuando las variables se introducen mediante una directiva del compilador, reciben un valor por defecto, que mantendrán durante la sesión mientras no hayan sido asignadas.

El valor por defecto depende del tipo de variable:

Tipo	Valor por defecto
Booleano	False
Fecha	00-00-00
Entero largo	0
Hora	00:00:00
Imagen	picture size=0
Real	0
Puntero	Nil=true
Texto	""
Blob	Tamaño Blob=0
Objeto	null
Collection	null
Variant	indefinido

## Convertir los tipos de datos

El lenguaje 4D contiene operadores y comandos para convertir entre tipos de datos, cuando dichas conversiones tienen sentido. El lenguaje 4D aplica la verificación de tipos de datos. Por ejemplo, no se puede escribir: "abc"+0.5+!12/25/96!-?00:30:45?. Esto generará errores de sintaxis.

La siguiente tabla lista los tipos de datos básicos, los tipos de datos a los que se pueden convertir y los comandos utilizados para hacerlo:

Tipos a convertir	en Cadena	en Número	en Fecha	en Hora	en Booleano
String (1)		Num	Fecha	Hora	Bool
Número (2)	Cadena				Bool
Fecha	Cadena				Bool
Hora	Cadena				Bool
Booleano		Num			

(1) Las cadenas formateadas en JSON pueden convertirse en datos escalares, objetos o colecciones, utilizando el comando `JSON Parse`.

(2) Los valores de tipo Hora pueden tratarse como números.

Nota: además de las conversiones de datos listadas en esta tabla, se pueden obtener conversiones de datos más sofisticadas combinando operadores y otros comandos.

# BLOB

Un campo, variable o expresión BLOB (Binary Large OBject) es una serie contigua de bytes que puede ser tratada como un objeto completo o cuyos bytes pueden ser direccionados individualmente.

Un blob se carga en la memoria en su totalidad. Una variable blob se mantiene y existe sólo en la memoria. Un campo blob se carga en memoria desde el disco, como el resto del registro al que pertenece.

Al igual que otros tipos de campo que pueden retener una gran cantidad de datos (como el tipo de campo Imagen), los campos blob no se duplican en la memoria cuando se modifica un registro. Por consiguiente, el resultado devuelto por los comandos `Old` y `Modified` no es significativo cuando se aplica a un campo blob.

## Tipos Blob

Utilizando el lenguaje 4D, hay dos maneras de manipular un blob:

- como un valor escalar: un blob puede ser almacenado en una variable o un campo Blob y puede ser modificado.
- como un objeto (`4D.Blob`): un `4D.Blob` es un objeto blob. Puede encapsular un blob o una parte de él en un `4D.Blob` sin alterar el bloque original. Este método se llama [boxing](#). Para más información sobre cómo instanciar un `4D.Blob`, vea [Blob Class](#).

Cada tipo de blob tiene sus ventajas. Utilice la siguiente tabla para determinar cuál se ajusta a sus necesidades:

	Blob	4D.Blob
Alterable	Sí	No
Compartible en objetos y colecciones	No	Sí
Pasado por referencia*	No	Sí
Rendimiento al acceder a los bytes	+	-
Tamaño máximo	2GB	Memoria

\*A diferencia de los comandos 4D diseñados para tomar un blob escalar como parámetro, pasar un blob escalar a un método duplicarlo en memoria. Al trabajar con métodos, usar objetos blob (`4D.Blob`) es más eficiente, ya que son pasados por referencia.

Por defecto, 4D define el tamaño máximo de los blobs escalares en 2GB, pero este límite de tamaño puede ser menor dependiendo de su sistema operativo y del espacio disponible.

No se pueden utilizar operadores en los blobs.

## Verificar si una variable contiene un blob escalar o un `4D.Blob`

Utilice el comando [Value type](#) para determinar si un valor es de tipo Blob u Objeto. Para verificar que un objeto es un objeto blob (`4D.Blob`), utilice [instancia OB de](#):

```
var $myBlob: Blob
var $myBlobObject: 4D.Blob
$myBlobObject:=4D.Blob.new()

$type:= Value type($myblobObject) // 38 (object)
$is4DBlob:= OB Instance of($myblobObject; 4D.Blob) //True
```

## Pasar blobs como parámetros

Los bloques escalares y los objetos blob pueden pasarse como parámetros a los comandos 4D o a las rutinas de plug-in que esperan parámetros blob.

### Pasar blobs y objetos blob a los comandos 4D

Puede pasar un blob escalar o un `4D.Blob` a todo comando 4D que tome un blob como parámetro:

```
var $myBlob: 4D.Blob  
CONVERT FROM TEXT("Hello, World!"; "UTF-8"; $myBlob)  
$myText:= BLOB to text( $myBlob ; UTF8 text without length )
```

Algunos comandos 4D modifican el blob, y por lo tanto no soportan el tipo `4D.Blob`:

- [DELETE FROM BLOB](#)
- [INSERT IN BLOB](#)
- [INTEGER TO BLOB](#)
- [LONGINT TO BLOB](#)
- [REAL TO BLOB](#)
- [SET BLOB SIZE](#)
- [TEXT TO BLOB](#)
- [VARIABLE TO BLOB](#)
- [LIST TO BLOB](#)
- [SOAP DECLARATION](#)
- [WEB SERVICE SET PARAMETER](#)

### Pasar blobs y objetos blob a los métodos

Puede pasar blobs y objetos blob (`4D.Blob`) a los métodos. Tenga en cuenta que a diferencia de los objetos blob, que son pasados por referencia, los blobs escalares se duplican en la memoria cuando se pasan a los métodos.

#### Pasar un blob escalar por referencia usando un puntero

Pasar un blob escalar a sus propios métodos sin duplicarlo en memoria, defina un puntero a la variable que lo almacena y pase el puntero como parámetro.

Ejemplos:

```
// Declarar una variable de tipo Blob  
var $myBlobVar: Blob  
// Pasar el blob como parámetro a un comando 4D  
SET BLOB SIZE($myBlobVar; 024*1024)
```

```
// Pasar el blob como parámetro a una rutina externa  
$errCode:=Hacer algo con este blob($myBlobVar)
```

```
// Pasar el blob como un parámetro a un método que devuelve un blob  
var $retrieveBlob: Blob  
retrieveBlob:=Fill_Blob($myBlobVar)
```

```
// Pasa un puntero al blob como parámetro a su propio método,  
COMPUTE BLOB(->$myBlobVar)
```

Nota para los desarrolladores de plugins: un parámetro BLOB se declara como "&O" (la letra "O", no el dígito "0").

## Asignar una variable Blob a otra

Puede asignar una variable Blob a otra:

Ejemplo:

```
// Declarar dos variables de tipo Blob
var $vBlobA; $vBlobB : Blob
// Establecer el tamaño del primer blob en 10K
SET BLOB SIZE($vBlobA; 0*1024)
// Asignar el primer blob al segundo
$vBlobB:=$vBlobA
```

## Conversión automática del tipo blob

4D convierte automáticamente los blobs escalares a objetos blob y viceversa, cuando se les asigna uno al otro. Por ejemplo:

```
// Crear una variable de tipo Blob y una variable objeto
var $myBlob: Blob
var $myObject : Objeto

// Asignar ese blob a una propiedad de $myObject llamada "blob"
$myObject:=New object("blob"; $myBlob)

// El blob almacenado en $myBlob se convierte automáticamente en un 4D.Blob
$type:= OB Instance of($myObject.blob; 4D.Blob) //True

// Conversión de un 4D.Blob en Blob
$myBlob:= $myObject.blob
$type:= Value type($myBlob) // Blob
```

Al convertir un `4D.Blob` a un blob escalar, si el tamaño del `4D.Blob` excede el tamaño máximo para los blobs escalares, el blob escalar resultante está vacío. Por ejemplo, cuando el tamaño máximo para los blobs escalares es 2GB, si convierte un `4D.Blob` de 2,5GB a un blob, obtiene un blob vacío.

## Modificación de un blob escalar

A diferencia de los objetos blob, se pueden modificar los blobs escalares. Por ejemplo:

```
var $myBlob : Blob
SET BLOB SIZE ($myBlob ; 16*1024)
```

## Acceder individualmente a los bytes de un blob

Acceder a los bytes de un blob escalar

Puede acceder a los bytes individuales de un blob escalar utilizando las llaves `{}`. Dentro de un blob, los bytes se numeran de 0 a N-1, donde N es el tamaño del BLOB:

```
// Declarar una variable de tipo Blob
var $vBlob : Blob
// Establecer el tamaño del blob en 256 bytes
SET BLOB SIZE($vBlob; 56)
// El siguiente código hace un bucle a través del blob para inicializar cada byte en cero
For(vByte; ;Tamaño BLOB ($vBlob)-1)
    $vBlob{vByte}:=0
Fin por
```

Como puede dirigir todos los bytes de un blob individualmente, puede almacenar lo que desee en una variable o un campo Blob.

#### Acceder a los bytes de un `4D.Blob`

Utilice los corchetes `[]` para acceder directamente a un byte específico en un `4D.Blob`

```
var $myBlob: 4D.Blob
CONVERT FROM TEXT("Hello, World!"; "UTF-8"; $myBlob)
$myText:= BLOB to text ( $myBlob ; UTF8 text without length )
$byte:=$myBlob[5]
```

Dado que un `4D.Blob` no puede ser modificado, puede leer los bytes de un `4D.Blob` utilizando esta sintaxis, pero no modificarlos.

# Booleano

Un campo, variable o expresión booleana puede ser TRUE o FALSE.

## Funciones booleanas

4D ofrece las funciones booleanas `True`, `False` y `Not` en el tema dedicado Booleanos. Para más información, consulte las descripciones de estos comandos.

### Ejemplo

Este ejemplo define una variable booleana basada en el valor de un botón. Devuelve True en `myBoolean` si el botón `myButton` fue presionado y False si el botón no fue presionado. Cuando se hace clic en un botón, la variable del botón toma el valor 1.

```
If(myButton=1) //Si se ha presionado el botón  
    myBoolean:=True //myBoolean toma el valor True  
Else //Si el botón no fue pulsado  
    myBoolean:=False //myBoolean toma el valor False  
End if
```

El ejemplo anterior puede simplificarse en una línea.

```
myBoolean:=(myButton=1)
```

## Operadores lógicos

4D soporta dos operadores lógicos que trabajan sobre expresiones booleanas: la conjunción (AND) y la disyunción inclusiva (OR). Un AND lógico devuelve TRUE si ambas expresiones son TRUE. Un OR lógico devuelve TRUE si al menos una de las expresiones es TRUE. La siguiente tabla muestra los operadores lógicos:

Operación	Sintaxis	Devuelve	Expresión	Valor
AND	Booleano & Booleano	Booleano	("A" = "A") & (15 # 3)	True
			("A" = "B") & (15 # 3)	False
			("A" = "B") & (15 = 3)	False
OR	Booleano   Booleano	Booleano	("A" = "A")   (15 # 3)	True
			("A" = "B")   (15 # 3)	True
			("A" = "B")   (15 = 3)	False

La siguiente es la tabla de verdad del operador lógico AND:

Expr1	Expr2	Expr1 & Expr2
True	True	True
True	False	False
False	True	False
False	False	False

La siguiente es la tabla de verdad del operador lógico OR:

Expr1	Expr2	Expr1   Expr2
True	True	True
True	False	True
False	True	True
False	False	False

Consejo: si necesita calcular la conjunción exclusiva entre Expr1 y Expr2, escriba:

```
(Expr1|Expr2) & Not(Expr1 & Expr2)
```

En contextos booleanos, el lenguaje 4D también soporta los operadores [cortocircuitos](#) (`&&` y `||`) y el concepto [truthy y falsy](#).

# Collection

Las colecciones son listas ordenadas de valores de tipos similares o diferentes (texto, número, fecha, objeto, booleano, colección o null).

Para manipular las variables de tipo Colección, debe utilizar la notación objeto (ver [Sintaxis-básica](#)).

Para acceder a un elemento de la colección, hay que pasar el número del elemento entre corchetes:

```
collectionRef[expression]
```

Puede pasar toda expresión 4D válida que devuelva un entero positivo en *expresión*. Ejemplos:

```
myCollection[5] //acceso al 6º elemento de la colección  
myCollection[$var]
```

Atención: los elementos de la colección están numerados desde 0.

Puede asignar un valor a un elemento de la colección u obtener el valor de un elemento de colección:

```
myCol[10]:="My new element"  
$myVar:=myCol[0]
```

Si se asigna un índice de elemento que sobrepasa el último elemento existente de la colección, la colección se redimensiona automáticamente y a todos los nuevos elementos intermedios se les asigna un valor nulo:

```
var myCol : Collection  
myCol:=New collection("A";"B")  
myCol[5]:="Z"  
//myCol[2]=null  
//myCol[3]=null  
//myCol[4]=null
```

## Inicialización

Las colecciones deben haber sido inicializadas, por ejemplo utilizando el comando `New collection`, de lo contrario al intentar leer o modificar sus elementos se generará un error de sintaxis.

Ejemplo:

```
var $colVar : Collection //creación de una variable 4D de tipo colección  
$colVar:=New collection //inicialización de la colección y asignación a la variable 4D
```

## Colección estándar o compartida

Puede crear dos tipos de colecciones:

- colecciones estándar (no compartidas), utilizando el comando `New collection`. Estas colecciones pueden ser editadas sin ningún control de acceso específico, pero no pueden ser compartidas entre procesos.
- colecciones compartidas, utilizando el comando `New shared collection`. Estas colecciones pueden ser compartidas entre procesos, incluidos los hilos apropiativos. El acceso a estas colecciones se controla mediante estructuras `Use...End use`.

Para más información, consulte la sección [Objetos y colecciones compartidos](#).

## Funciones de colección

Las referencias a colecciones 4D se benefician de funciones de clase específicas (a veces llamados *funciones métodos*). Las funciones de colección se listan en la sección [Class API Reference](#).

Por ejemplo:

```
$newCol:=$col.copy() //copia de $col a $newCol  
$col.push(10;100) //añade de 10 y 100 a la colección
```

Ciertas funciones devuelven la colección original después de la modificación, para que pueda ejecutar las llamadas en una secuencia:

```
$col:=New collection(5;20)  
$col2:=$col.push(10;100).sort() // $col2=[5,10,20,100]
```

### parámetro rutaPropiedad

Varias funciones aceptan una *propertyPath* como parámetro. Este parámetro significa:

- o bien un nombre de propiedad del objeto, por ejemplo "apellido"
- o una ruta de propiedades del objeto, es decir, una secuencia jerárquica de subpropiedades vinculadas con caracteres de punto, por ejemplo "empleado.hijos.nombre".

Atención: cuando se utilizan funciones y parámetros *propertyPath*, no se puede utilizar ".", "[ ]", o espacios en los nombres de las propiedades ya que impedirá que 4D analice correctamente la ruta:

```
$vmin:=$col.min("My.special.property") //indefinido  
$vmin:=$col.min(["My.special.property"]) //error
```

# Fecha

Las variables, campos o expresiones de tipo fecha pueden estar comprendidas entre 1/1/100 y 31/12/32.767.

Aunque el modo de representación de fechas por C\_DATE permite trabajar con fechas hasta el año 32 767, ciertas operaciones que pasan por el sistema imponen un límite inferior.

Nota: en el manual de Referencia del Lenguaje 4D, los parámetros de tipo Fecha en las descripciones de los comandos se denominan Fecha, salvo que se indique lo contrario.

## Constantes literales de tipo fecha

Una constante literal de tipo fecha está rodeada de signos de exclamación (!...!). Una fecha debe estar estructurada utilizando el formato ISO (!AAAA-MM-DD!). Estos son algunos ejemplos de constantes de fechas:

```
!1976-01-01!
!2004-09-29!
!1976-01-01!
!2004-09-29!
!2015-12-31!
!2004-09-29!
!1976-01-01!
!2004-09-29!
!2015-12-31!
```

Una fecha nula es especificada por *!00-00-00!*.

Consejo: el Editor de métodos incluye un acceso directo para introducir una fecha nula. Para escribir una fecha nula, introduzca el carácter de exclamación (!) y pulse Intro.

Notas:

- Por razones de compatibilidad, 4D acepta que se introduzcan años de dos dígitos. Se asume que un año de dos dígitos se encuentra en el siglo XX o en el XXI según sea mayor o menor de 30, a menos que esta configuración por defecto se haya cambiado utilizando el comando SET DEFAULT CENTURY .
- Si ha marcado la opción "Utilizar la configuración regional del sistema" ( ver Página Métodos), debe utilizar el formato de fecha definido en su sistema. Generalmente, en un entorno estadounidense, las fechas se introducen en la forma mes/día/año, con una barra "/" que separa los valores.

## Operadores de fechas

Operación	Sintaxis	Devuelve	Expresión	Valor
Diferencia	Fecha – Fecha	Número	$!2017-01-20! - !2017-01-01!$	19
Adición	Fecha + Número	Fecha	$!2017-01-20! + 9$	$!2017-01-29!$
Resta	Fecha - Número	Fecha	$!2017-01-20! - 9$	$!2017-01-11!$
Igual	Fecha = Fecha	Booleano	$!2017-01-20! - !2017-01-01! = !2017-01-01!$	True
			$!2017-01-20! \neq !2017-01-20! \neq !2017-01-01! = !2017-01-01!$	False
Desigualdad	Fecha # Fecha	Booleano	$!2017-01-20! \neq !2017-01-20! \neq !2017-01-20! \# !2017-01-01!$	True
			$!2017-01-20! \neq !2017-01-20! \# !2017-01-20!$	False
Mayor que	Fecha > Fecha	Booleano	$!2017-01-20! \neq !2017-01-20! > !2017-01-01!$	True
			$!2017-01-20! > !2017-01-20!$	False
Menor que	Fecha < Fecha	Booleano	$!2017-01-20! - !2017-01-01! < !2017-01-20!$	True
			$!2017-01-20! < !2017-01-20!$	False
Mayor o igual que	Fecha $\geq$ Fecha	Booleano	$!2017-01-20! \neq !2017-01-20! \geq !2017-01-01!$	True
			$!2017-01-01! \geq !2017-01-20!$	False
Menor o igual que	Fecha $\leq$ Fecha	Booleano	$!2017-01-01! \leq !2017-01-20!$	True
			$!2017-01-20! \leq !2017-01-01!$	False

# Null e indefinido

Null e Indefinido son tipos de datos que manejan los casos en los que no se conoce el valor de una expresión.

## Null

Null es un tipo de datos especial con un solo valor posible: null. Este valor es devuelto por una expresión que no contiene ningún valor.

En el lenguaje 4D y para los atributos de los campos de los objetos, los valores nulos se gestionan a través de la función `Null`. Esta función puede utilizarse con las siguientes expresiones para definir o comparar el valor nulo:

- atributos de objetos
- elementos de colecciones
- variables de tipo objeto, colección, puntero, imagen o variante.

## Indefinido

Indefinido no es realmente un tipo de datos. Denota una variable que aún no ha sido definida. Una función (un método de proyecto que devuelve un resultado) puede devolver un valor indefinido si, dentro del método, se asigna al resultado de la función (\$0) una expresión indefinida (una expresión calculada con al menos una variable indefinida). Un campo no puede ser indefinido (el comando `Undefined` siempre devuelve False para un campo). Una variable variant tiene indefinido como valor por defecto.

## Ejemplos

Aquí están los diferentes resultados del comando `Undefined` así como del comando `Null` con las propiedades de los objetos, dependiendo del contexto:

```
C_OBJECT($vEmp)
$vEmp:=New object
$vEmp.name:="Smith"
$vEmp.children:=Null

$undefined:=Undefined($vEmp.name) // False
>null:=( $vEmp.name=NULL) //False

$undefined:=Undefined($vEmp.children) // False
>null:=( $vEmp.children=NULL) //True

$undefined:=Undefined($vEmp.parent) // True
>null:=( $vEmp.parent=NULL) //True
```

# Número (Real, Entero largo, Entero)

Número es un término genérico que significa:

- Los campos, variables o expresiones de tipo real. El rango del tipo Real es  $\pm 1,7e\pm 308$  (13 dígitos significativos).
- Los campos, variables o expresiones de tipo Entero largo. El rango para el tipo de datos Entero largo (4 bytes) es - $2^{31}..(2^{31}-1)$ .
- Los campos, variables o expresiones de tipo Entero. El rango para el tipo de datos Entero (2 bytes) es - $32.768..32.767$  ( $2^{15}..(2^{15}-1)$ ).

Nota: los valores de los campos enteros se convierten automáticamente en enteros largos cuando se utilizan en el lenguaje 4D.

Puede asignar cualquier tipo de dato numérico a otro; 4D realiza la conversión, truncando o redondeando si es necesario. Sin embargo, cuando los valores están fuera del rango, la conversión no devolverá un valor válido. Se pueden mezclar los tipos de datos numéricos en las expresiones.

Nota: en el manual de referencia del lenguaje 4D, sin importar el tipo de datos real, los parámetros de tipo Real, Entero y Entero largo en las descripciones de los comandos se indican como número, salvo que se indique lo contrario.

## Constantes literales numéricas

Una constante literal numérica se escribe como un número real. Estos son algunos ejemplos de constantes numéricas:

27  
123.76  
0.0076

El separador decimal por defecto es el punto (.), independientemente del lenguaje del sistema. Si ha marcado la opción "Utilizar la configuración regional del sistema" en la página de Métodos de las Preferencias, debe utilizar el separador definido en su sistema.

Los números negativos se especifican con el signo menos (-). Por ejemplo:

-27  
-123.76  
-0.0076

## Operadores numéricos

Operación	Sintaxis	Devuelve	Expresión	Valor
Adición	Número + Número	Número	2 + 3	5
Resta	Número - Número	Número	3 - 2	1
Multiplicación	Número * Número	Número	5 * 2	10
División	Número / Número	Número	5 / 2	2.5
División entera	Número ¥ Número	Número	5 ¥ 2	2
Módulo	Número % Número	Número	5 % 2	1
Exponenciación	Número ^ Número	Número	2 ^ 3	8
Igual	Número = Número	Booleano	10 = 10	True
			10 = 11	False
Desigualdad	Número # Número	Booleano	10 # 11	True
			10 # 10	False
Mayor que	Número > Número	Booleano	11 > 10	True
			10 > 11	False
Menor que	Número < Número	Booleano	10 < 11	True
			11 < 10	False
Mayor o igual que	Número >= Número	Booleano	11 >= 10	True
			10 >= 11	False
Menor o igual que	Número <= Número	Booleano	10 <= 11	True
			11 <= 10	False

El operador modulo % divide el primer número entre el segundo y devuelve un resto de número entero. He aquí algunos ejemplos:

- 10 % 2 devuelve 0 porque 10 está dividido uniformemente por 2.
- 10 % 3 devuelve 1 porque el resto es 1.
- 10,5 % 2 devuelve 0 porque el resto no es un número entero.

#### ATENCIÓN:

- El operador modulo % devuelve valores significativos con números que están en el rango de los enteros largos (de  $-2^{31}$  hasta  $2^{31}$  menos 1). Para calcular el módulo con números fuera de este rango, utilice el comando `Mod`.
- El operador de división entero largo ¥ devuelve valores significativos sólo con números enteros.

## Prioridad

El orden en que se evalúa una expresión se llama prioridad. 4D tiene una precedencia estricta de izquierda a derecha, en la que no se aplica el orden algebraico. Por ejemplo:

3+4\*5

devuelve 35, porque la expresión se evalúa como  $3 + 4$ , dando como resultado 7, que luego se multiplica por 5, con el resultado final de 35.

Para anular la precedencia de izquierda a derecha, DEBE utilizar paréntesis. Por ejemplo:

3+(4\*5)

devuelve 23 porque la expresión `(4 * 5)` se evalúa primero, debido a los paréntesis. El resultado es 20, que se suma a 3 para el resultado final de 23.

Los paréntesis pueden anidarse dentro de otros conjuntos de paréntesis. Asegúrese de que cada paréntesis de la izquierda tenga un paréntesis de la derecha que coincida para garantizar la evaluación correcta de las expresiones. La falta o el uso incorrecto de los paréntesis puede provocar resultados inesperados o expresiones no válidas. Además, si pretende compilar sus aplicaciones, debe tener paréntesis coincidentes: el compilador detecta la falta de paréntesis como un error de sintaxis.

## Operadores de bits

Los operadores de bits operan sobre expresiones o valores Entero largo.

Si se pasa un valor de tipo Entero o Real a un operador de tipo bit, 4D evalúa el valor como un valor de tipo Entero Largo antes de calcular el resultado de la expresión.

Cuando se utilizan los operadores de bits, hay que pensar en un valor de tipo Entero largo como un array de 32 bits. Los bits están numerados de 0 a 31, de derecha a izquierda.

Dado que cada bit puede ser igual a 0 o 1, también se puede pensar en un valor Entero largo como un valor en el que se pueden almacenar 32 valores booleanos. Un bit igual a 1 significa Verdadero y un bit igual a 0 significa Falso.

Una expresión que utiliza un operador bitwise devuelve un valor Entero largo, excepto para el operador Bit Test, donde la expresión devuelve un valor Booleano. La siguiente tabla lista los operadores a nivel de bits y su sintaxis:

Operación	Operador	Sintaxis	Devuelve
Y	&	Long & Long	Long
O (inclusive)		Long   Long	Long
O (exclusivo)	^	Long ^  Long	Long
Left Bit Shift	<<	Long << Long	Long (ver nota 1)
Right Bit Shift	>>	Long >> Long	Long (ver nota 1)
Bit Set	?+	Long ?+ Long	Long (ver nota 2)
Poner el bit en 0	?-	Long ?- Long	Long (ver nota 2)
Probar bit	??	Long ?? Long	Boolean (ver nota 2)

### Notas

- Para las operaciones `Left Bit Shift` and `Right Bit Shift`, el segundo operando indica el número de posiciones en que se desplazarán los bits del primer operando en el valor resultante. Por lo tanto, este segundo operando debe estar entre 0 y 31. Tenga en cuenta, sin embargo, que el desplazamiento de 0 devuelve un valor sin cambios y el desplazamiento de más de 31 bits devuelve 0x00000000 porque todos los bits se pierden. Si se pasa otro valor como segundo operando, el resultado no es significativo.
- En las operaciones `Bit Set`, `Bit Clear` y `Bit Test`, el segundo operando indica el número del bit sobre el que hay que actuar. Por lo tanto, este segundo operando debe estar entre 0 y 31; de lo contrario, el resultado de la expresión no es significativo.

La siguiente tabla lista los operadores a nivel de bits y sus efectos:

Operación	Descripción
Y	<p>Cada bit resultante es el resultado de la operación AND lógica aplicada a los bits de los dos operandos. Aquí está la tabla del AND lógico:</p> <ul style="list-style-type: none"> <li>• 1 &amp; 1 --&gt; 1</li> <li>• 0 &amp; 1 --&gt; 0</li> <li>• 1 &amp; 0 --&gt; 0</li> <li>• 0 &amp; 0 --&gt; 0</li> </ul> <p>Es decir, el bit resultante es 1 si los dos bits del operando son 1; en caso contrario, el bit resultante es 0.</p>
O (inclusive)	<p>Cada bit resultante es el resultado de la operación OR lógica aplicada a los dos bits operandos. Aquí está la tabla del OR lógico:</p> <ul style="list-style-type: none"> <li>• 1   1 --&gt; 1</li> <li>• 0   1 --&gt; 1</li> <li>• 1   0 --&gt; 1</li> <li>• 0   0 --&gt; 0</li> </ul> <p>Es decir, el bit resultante es 1 si al menos uno de los dos bits del operando es 1; en caso contrario, el bit resultante es 0.</p>
O (exclusivo)	<p>Cada bit resultante es el resultado de la operación XOR lógica aplicada a los dos bits operandos. Aquí está la tabla del XOR lógico:</p> <ul style="list-style-type: none"> <li>• 1 ^  1 --&gt; 0</li> <li>• 0 ^  1 --&gt; 1</li> <li>• 1 ^  0 --&gt; 1</li> <li>• 0 ^  0 --&gt; 0</li> </ul> <p>Es decir, el bit resultante es 1 si solo uno de los dos bits del operando es 1; en caso contrario, el bit resultante es 0.</p>
Left Bit Shift	<p>El valor resultante se ajusta al valor del primer operando, luego los bits resultantes se desplazan a la izquierda el número de posiciones indicado por el segundo operando. Los bits de la izquierda se pierden y los nuevos bits de la derecha se ponen en 0.</p> <p>Nota: teniendo en cuenta sólo los valores positivos, desplazar a la izquierda N bits es lo mismo que multiplicar por <math>2^N</math>.</p>
Right Bit Shift	<p>El valor resultante se ajusta al valor del primer operando, luego los bits resultantes se desplazan a la derecha el número de posición indicado por el segundo operando. Los bits de la derecha se pierden y los nuevos bits de la izquierda se ponen en 0.</p> <p>Nota: teniendo en cuenta sólo los valores positivos, desplazar a la derecha N bits es lo mismo que dividir por <math>2^N</math>.</p>
Bit Set	El valor resultante se establece en el valor del primer operando, luego el bit resultante, cuyo número es indicado por el segundo operando, se coloca en 1. Los demás bits no se modifican.
Poner el bit en 0	El valor resultante se establece en el valor del primer operando, luego el bit resultante, cuyo número es indicado por el segundo operando, se coloca en 0. Los demás bits no se modifican.
Probar bit	Devuelve True si, en el primer operando, el bit cuyo número indica el segundo operando es igual a 1. Devuelve False si, en el primer operando, el bit cuyo número indica el segundo operando es igual a 0.

## Ejemplos

Operación	Ejemplo	Resultado
Y	0x0000FFFF & 0xFF00FF00	0x0000FF00
O (inclusive)	0x0000FFFF   0xFF00FF00	0xFF00FFFF
O (exclusivo)	0x0000FFFF ^  0xFF00FF00	0xFF0000FF
Left Bit Shift	0x0000FFFF << 8	0x00FFFF00
Right Bit Shift	0x0000FFFF >> 8	0x000000FF
Bit Set	0x00000000 ?+ 16	0x00010000
Poner el bit en 0	0x00010000 ?- 16	0x00000000
Probar bit	0x00010000 ?? 16 16 16 16	True

# Objeto

Las variables, campos o expresiones de tipo objeto pueden contener datos de diversos tipos. La estructura de los objetos "nativos" 4D se basa en el principio clásico de los pares "propiedad/valor". La sintaxis de estos objetos se basa en la notación JSON:

- El nombre de una propiedad es siempre un texto, por ejemplo "Name". Debe seguir [reglas específicas](#).
- Un valor de propiedad puede ser del tipo siguiente:
  - número (Real, Entero, etc.)
  - texto
  - null
  - booleano
  - puntero (almacenado como tal, evaluado con el comando `JSON Stringify` o al copiar),
  - fecha (tipo fecha o cadena en formato fecha ISO)
  - objeto(1) (los objetos pueden ser anidados en varios niveles)
  - imagen(2)
  - colección

(1) Los objetos ORDA como [entidades](#) o las [selecciones de entidades](#) no pueden almacenarse en campos objeto; sin embargo, se soportan completamente en las variables objeto en memoria.

(2) Cuando se expone como texto en el depurador o se exporta a JSON, las propiedades de los objetos de tipo imagen indican "[objeto Imagen]".

Atención: recuerde que los nombres de atributos diferencian entre mayúsculas y minúsculas.

Las variables, campos o expresiones de tipo Objeto se gestionan utilizando la [notación objeto](#) o los comandos clásicos disponibles en el tema Objetos (Lenguaje). Tenga en cuenta que se pueden utilizar comandos específicos del tema Búsquedas, como `QUERY BY ATTRIBUTE`, `QUERY SELECTION BY ATTRIBUTE`, o `ORDER BY ATTRIBUTE` para llevar a cabo el procesamiento de los campos objetos.

Cada valor de propiedad al que se accede a través de la notación de objeto se considera una expresión. Puede utilizar estos valores siempre que se esperen expresiones 4D:

- en código 4D, ya sea escrito en los métodos (editor de métodos) o externalizado (fórmulas, archivos de etiquetas procesados por `PROCESS 4D TAGS` o el servidor web, archivos de exportación, documentos 4D Write Pro...),
- en las áreas de expresiones del depurador y del explorador de ejecución,
- en la lista de propiedades del editor de formularios para los objetos formulario: campo Variable o Expresión, así como diversas expresiones de list box y columnas (fuente de datos, color de fondo, estilo o color de fuente).

## Inicialización

Los objetos deben haber sido inicializados, por ejemplo utilizando el comando `New object`, de lo contrario al intentar leer o modificar sus propiedades se generará un error de sintaxis.

Ejemplo:

```
C_OBJECT($obVar) //creación de una variable 4D de tipo objeto  
$obVar:=New object //inicialización del objeto y asignación a la variable 4D
```

## Objeto estándar o compartido

Puede crear dos tipos de objetos:

- objetos estándar (no compartidos), utilizando el comando `New object`. Estos objetos pueden ser editados sin ningún control de acceso específico, pero no pueden ser compartidos entre procesos.

- objetos compartidos, utilizando el comando `New shared object`. Estos objetos pueden ser compartidos entre procesos, incluidos los hilos apropiativos. El acceso a estos objetos se controla mediante estructuras `Use...End use`. Para más información, consulte la sección [Objetos y colecciones compartidos](#).

## Principios de la sintaxis

La notación de objetos puede utilizarse para acceder a los valores de las propiedades de objetos a través de una cadena de tokens.

### Propiedades de los objetos

Con la notación de objetos, se puede acceder a las propiedades de los objetos de dos maneras:

- utilizando un símbolo "punto": > `object.propertyName`

Ejemplo:

```
employee.name:="Smith"
```

- utilizando una cadena entre corchetes: > `object["propertyName"]`

Ejemplos:

```
$vName:=employee["name"]
//o también:
$property:="name"
$vName:=employee[$property]
```

Como el valor de una propiedad de objeto puede ser un objeto o una colección, la notación objeto acepta una secuencia de símbolos para acceder a subpropiedades, por ejemplo:

```
$vAge:=employee.children[2].age
```

La notación de objetos está disponible en cualquier elemento del lenguaje que pueda contener o devolver un objeto, es decir:

- con los Objetos mismos (almacenados en variables, campos, propiedades de objetos, arrays de objetos o elementos de colecciones). Ejemplos:

```
$age:=$myObjVar.employee.age //variable
$addr:=[Emp]data_obj.address //campo
$city:=$addr.city //propiedad de un objeto
$pop:=$aObjCountries{2}.population //array de objetos
$val:=$myCollection[3].subvalue //elemento de colección
```

- comandos 4D que devuelven objetos. Ejemplo:

```
$measures:=Get database measures. DB.tables
```

- Métodos proyecto que devuelven objetos. Ejemplo:

```
// MyMethod1  
C_OBJECT($0)  
$0:=New object("a";10;"b";20)  
  
//myMethod2  
$result:=MyMethod1.a //10
```

- Collections Ejemplo:

```
myColl.length //tamaño de la colección
```

## Punteros

Nota preliminar: dado que los objetos se pasan siempre por referencia, no suele ser necesario utilizar punteros. Al pasar el objeto, internamente 4D utiliza automáticamente un mecanismo similar a un puntero, minimizando la necesidad de memoria y permitiendo modificar el parámetro y devolver las modificaciones. Como resultado, no es necesario utilizar punteros. Sin embargo, en caso de querer utilizar punteros, se puede acceder a los valores de las propiedades mediante punteros.

El uso de la notación de objetos con punteros es muy similar al uso de la notación de objetos directamente con objetos, excepto que el símbolo "punto" debe omitirse.

- Acceso directo:

```
pointerOnObject->propertyName
```

- Acceso por nombre:

```
pointerOnObject->["propertyName"]
```

Ejemplo:

```
C_OBJECT(v0bj)  
C_POINTER(vPtr)  
v0bj:=New object  
v0bj.a:=10  
vPtr:=->v0bj  
x:=vPtr->a //x=10
```

## Valor Null

Cuando se utiliza la notación de objetos, se soporta el valor null con el comando Null. Este comando puede utilizarse para asignar o comparar el valor nulo a propiedades de objetos o a elementos de colecciones, por ejemplo:

```
myObject.address.zip:=Null  
If(myColl[2]=Null)
```

Para más información, consulte la descripción del comando `Null`.

## Valor indefinido

La evaluación de una propiedad de un objeto puede producir a veces un valor indefinido. Normalmente, al intentar leer o asignar expresiones indefinidas, 4D generará errores. Esto no ocurre en los siguientes casos:

- La lectura de una propiedad de un objeto o valor indefinido devuelve un indefinido; la asignación de un valor

indefinido a variables (excepto arrays) tiene el mismo efecto que llamar `CLEAR VARIABLE` con ellas:

```
C_OBJECT($o)
C_LONGINT($val)
$val:=10 //val=10
$val:=$o.a //$. es indefinido (no hay error), y la asignación de este valor borra la variable
//val=0
```

- La lectura de la propiedad length de una colección indefinida produce 0:

```
C_COLLECTION($c) //variable creada pero no se ha definido ninguna colección
$size:=$c.length //size = 0
```

- Un valor indefinido pasado como parámetro a un método proyecto se convierte automáticamente en 0 o "" según el tipo de parámetro declarado.

```
C_OBJECT($o)
mymethod($o.a) //pasa un parámetro indefinido

//En el método mymethod
C_TEXT($1) //Parámetro de tipo texto
// $1 contiene ""
```

- Una expresión de condición se convierte automáticamente en falsa cuando se evalúa a indefinido con las palabras clave If y Case of:

```
C_OBJECT($o)
If($o.a) // false
End if
Case of
    :($o.a) // false
End case
```

- La asignación de un valor indefinido a una propiedad de objeto existente reinicializa o borra su valor, dependiendo de su tipo:
- Objeto, colección, puntero: Null
- Imagen: imagen vacía
- Booleano: False
- Cadena: ""
- Número: 0
- Fecha: !00-00-00! si la opción "Utilizar el tipo fecha en lugar del formato fecha ISO en los objetos" está activada, de lo contrario ""
- Hora: 0 (número de ms)
- Indefinido, Null: sin cambios

```
C_OBJECT($o)
$o:=New object("a";2)
$o.a:=$o.b //$.a=0
```

- La asignación de un valor indefinido a una propiedad de objeto no existente no hace nada.

Cuando se esperan expresiones de un tipo determinado en su código 4D, puede asegurarse de que tienen el tipo correcto incluso cuando se evalúan como indefinidas, rodeándolas con el comando de transformación 4D apropiado: `String`, `Num`, `Date`, `Time`, `Bool`. Estos comandos devuelven un valor vacío del tipo especificado cuando la expresión se evalúa como indefinida. Por ejemplo:

```
$myString:=Lowercase(String($o.a.b)) //asegurarse de obtener un valor de cadena aunque sea indefinido  
//para evitar errores en el código
```

## Ejemplos

La utilización de la notación de objetos simplifica el código 4D en el manejo de los mismos. Sin embargo, tenga en cuenta que la notación basada en comandos sigue siendo totalmente soportada.

- Escritura y lectura de propiedades de objetos (este ejemplo compara la notación de objetos y la notación de comandos):

```
// Utilizando la notación de objeto  
C_OBJECT($myObj) //declaración de una variable objeto 4D  
$myObj:=New object //crea un objeto y lo asigna a la variable  
$myObj.age:=56  
$age:=$myObj.age //56  
  
// Usando la notación por comando  
C_OBJECT($myObj2) //declara una variable objeto 4D  
OB SET($myObj2;"age";42) //crea un objeto y añade la propiedad age  
$age:=OB Get($myObj2;"age") //42  
  
// Por supuesto, se pueden mezclar ambas notaciones  
C_OBJECT($myObj3)  
OB SET($myObj3;"age";10)  
$age:=$myObj3.age //10
```

- Creación de propiedades y asignación de valores, incluyendo objetos:

```
C_OBJECT($Emp)  
$Emp:=New object  
$Emp.city:="London" //crea la propiedad city con el valor "London"  
$Emp.city:="Paris" //modifica la propiedad city  
$Emp.phone:=New object("office";"123456789";"home";"0011223344")  
//crea la propiedad phone y define su valor para un objeto
```

- Obtener un valor en un subobjeto es muy sencillo utilizando la notación de objetos:

```
$vCity:=$Emp.city //"Paris"  
$vPhone:=$Emp.phone.home //"0011223344"
```

- Puede acceder a las propiedades como cadenas utilizando el operador [ ]

```
$Emp["city"]:= "Berlin" //modifica la propiedad city  
//esto puede ser útil para crear propiedades a través de variables  
C_TEXT($addr)  
$addr:="address"  
For($i;1;4)  
    $Emp[$addr+String($i)]:="  
End for  
// crea 4 propiedades vacías "dirección1...dirección4" en el objeto $Emp
```

# Imagen

Un campo, variable o expresión de tipo Imagen puede ser cualquier imagen de Windows o Macintosh. En general, esto incluye toda imagen que pueda ser puesta en el portapapeles o leída desde el disco utilizando comandos 4D como `READ PICTURE FILE`.

4D utiliza APIs nativas para codificar (escribir) y decodificar (leer) los campos y las variables de las imágenes tanto en Windows como en macOS. Estas implementaciones dan acceso a numerosos formatos nativos, incluido el formato RAW, utilizado actualmente por las cámaras digitales.

- en Windows, 4D utiliza WIC (Windows Imaging Component).
- en macOS, 4D utiliza ImageIO.

WIC e ImageIO permiten el uso de metadatos en las imágenes. Dos comandos, `SET PICTURE METADATA` y `GET PICTURE METADATA`, le permiten beneficiarse de los metadatos en sus desarrollos.

## Identificadores de códecs de imágenes

4D soporta de forma nativa un amplio conjunto de [formatos de imagen](#), como .jpeg, .png o .svg.

Los formatos de imágenes reconocidos por 4D son devueltos por el comando `PICTURE CODEC LIST` como identificadores de códecs de imágenes. Se pueden devolver de las siguientes formas:

- Como una extensión (por ejemplo ".gif")
- Como un tipo MIME (por ejemplo, "image/jpeg")

La forma devuelta para cada formato dependerá de la forma en que se registre el códec a nivel del sistema operativo. Tenga en cuenta que la lista de códecs disponibles para lectura y escritura puede ser diferente, ya que los códecs de codificación pueden requerir licencias específicas.

La mayoría de los comandos de gestión de imágenes [4D](#) pueden recibir un Codec ID como parámetro. Por lo tanto, es imperativo utilizar el ID del sistema devuelto por el comando `PICTURE CODEC LIST`. Los formatos de imágenes reconocidos por 4D son devueltos por el comando `PICTURE CODEC LIST`.

## Operadores de imágenes

Operación	Sintaxis	Devuelve	Acción
Concatenación horizontal	Imagen1 + Imagen2	Imagen	Añadir Imagen2 a la derecha de Imagen1
Concatenación vertical	Imagen1 / Imagen2	Imagen	Añadir Imagen2 debajo de Imagen1
Superposición exclusiva	Imagen1 & Imagen2	Imagen	Superpone Imagen2 sobre Imagen1 (Imagen2 en primer plano). Produce el mismo resultado que <code>COMBINE PICTURES(pict3;pict1;Superimposition;pict2)</code>
Superposición inclusiva	Imagen1   Imagen2	Imagen	Superpone Imagen2 sobre Imagen1 y devuelve la máscara resultante si ambas imágenes tienen el mismo tamaño. Produce el mismo resultado que <code>\$equal:=Equal pictures(Pict1;Pict2;Pict3)</code>
Desplazamiento horizontal	Imagen + Número	Imagen	Mover la imagen horizontalmente un Número de píxeles
Movimiento vertical	Imagen / Número	Imagen	Mover la imagen verticalmente un Número de píxeles
Redimensionamiento	Imagen * Número	Imagen	Redimensiona la imagen según el porcentaje Número
Escala horizontal	Imagen *+ Número	Imagen	Redimensionar la imagen horizontalmente al porcentaje Número
Escala vertical	Imagen *  Número	Imagen	Redimensionar la imagen verticalmente al porcentaje Número

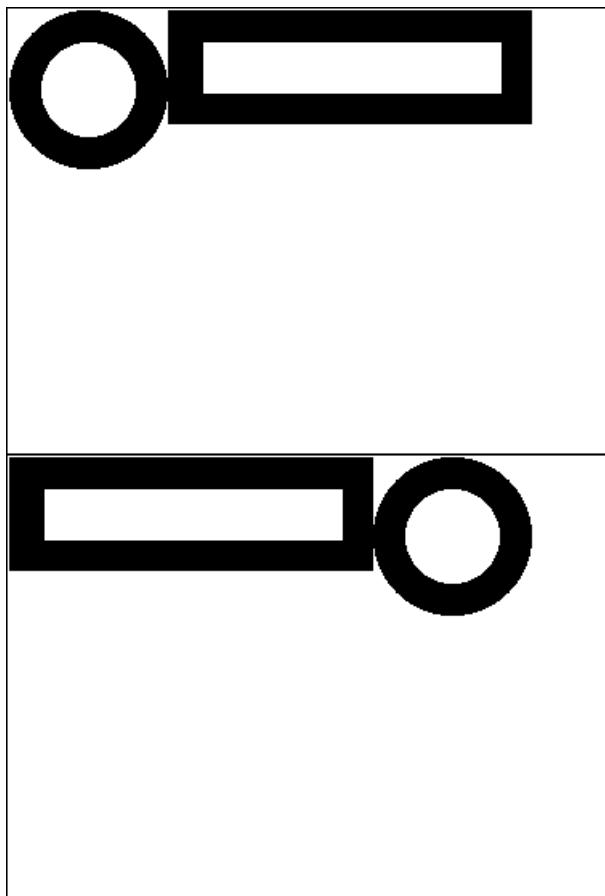
Notas:

- Para poder utilizar el operador |, Imag1 e Imag2 deben tener exactamente la misma dimensión. Si ambas imágenes tienen un tamaño diferente, la operación Imagen1 | Imagen2 produce una imagen en blanco.
- El comando `COMBINE PICTURES` puede utilizarse para superponer imágenes manteniendo las características de cada imagen fuente en la imagen resultante.
- Se pueden realizar operaciones adicionales en las imágenes utilizando el comando `TRANSFORM PICTURE`.
- No hay operadores de comparación de imágenes, sin embargo 4D propone el comando `Equal picture` para comparar dos imágenes.

## Ejemplos

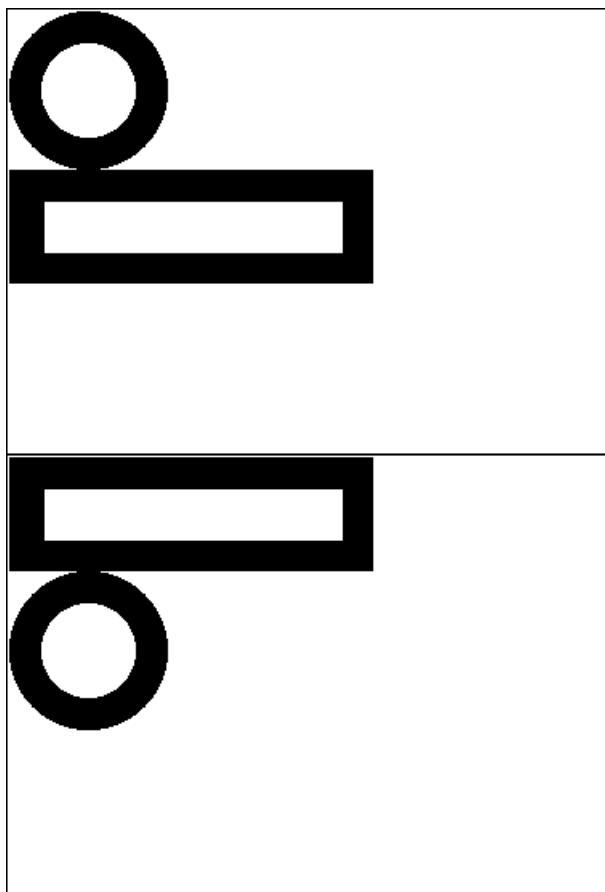
Concatenación horizontal

```
circle+rectangle //Colocar el rectángulo a la derecha del círculo
rectangle+circle //Colocar el círculo a la derecha del rectángulo
```



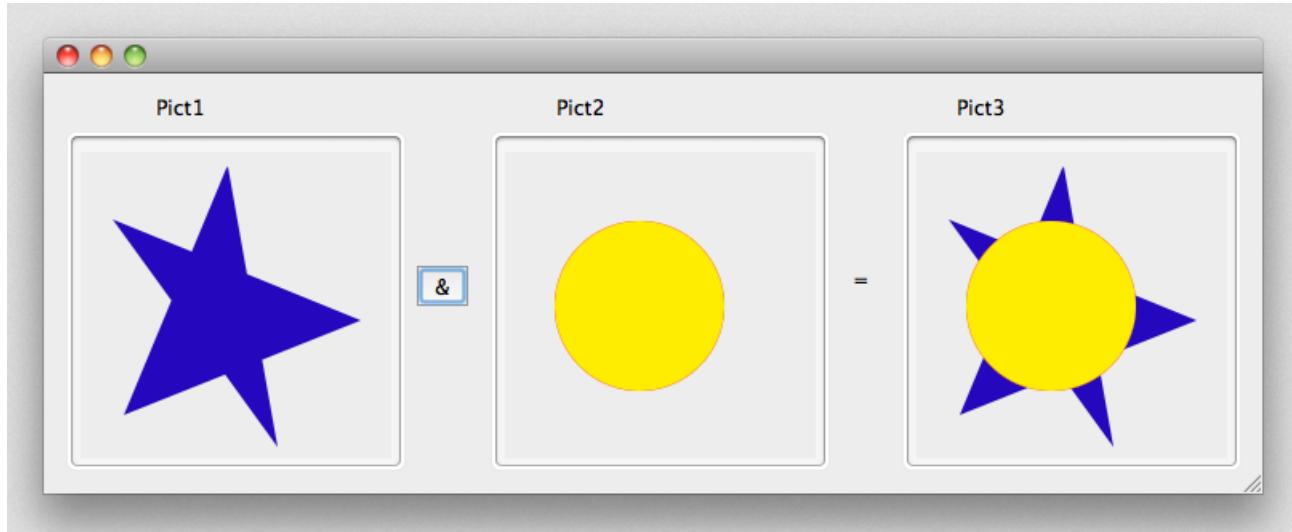
Concatenación vertical

```
circle+rectangle //Colocar el rectángulo debajo del círculo  
rectangle+circle //Colocar el círculo debajo del rectángulo
```



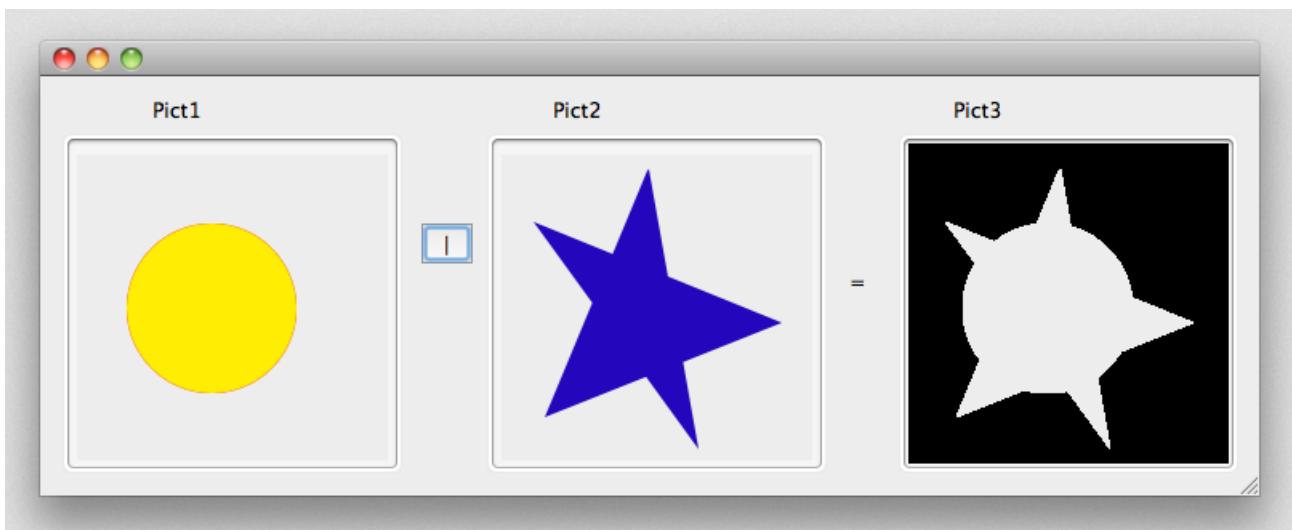
Superposición exclusiva

```
Pict3:=Pict1 & Pict2 // Superponer Pict2 sobre Pict1
```



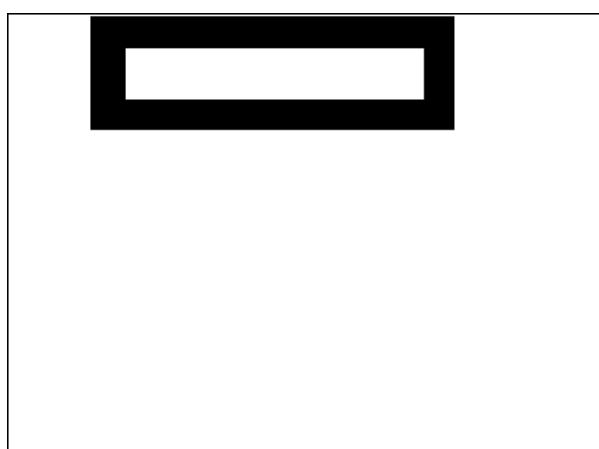
Superposición inclusiva

```
Pict3:=Pict1|Pict2 // Recupera la máscara resultante de la superposición de dos imágenes del mismo tamaño
```



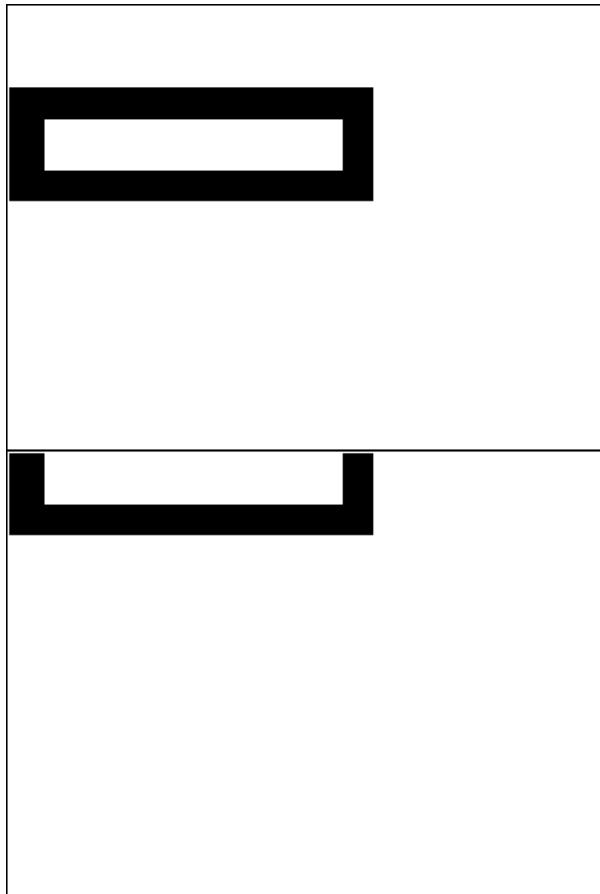
Desplazamiento horizontal

```
rectangle+50 //Mover el rectángulo 50 píxeles a la derecha  
rectangle-50 //Mover el rectángulo 50 píxeles a la izquierda
```



## Movimiento vertical

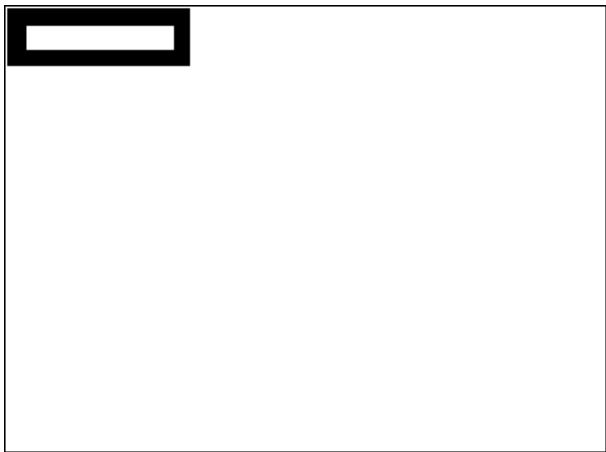
```
rectangle/50 //Mover el rectángulo 50 píxeles hacia abajo  
rectangle/-20 //Mover el rectángulo 20 píxeles hacia arriba
```



## Redimensionamiento

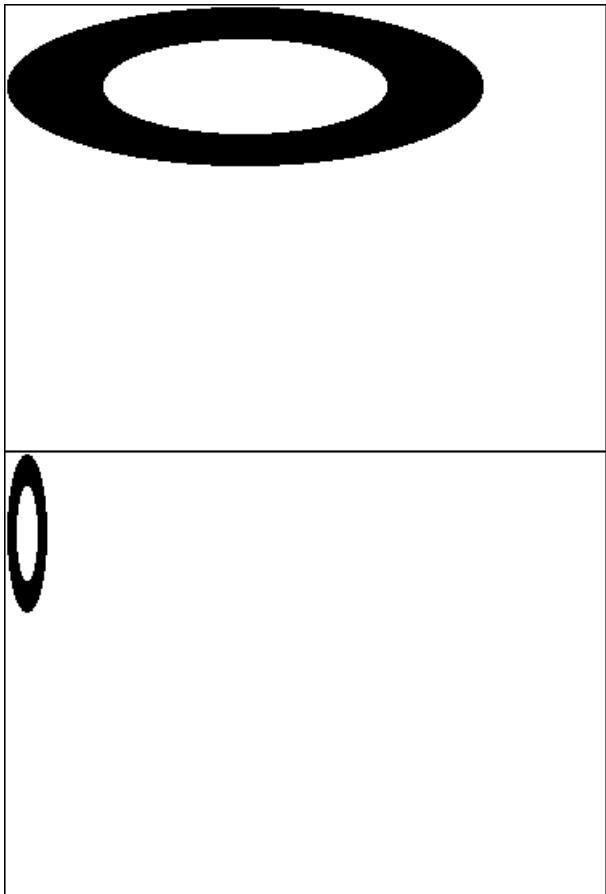
```
rectangle*1.5 //El rectángulo se hace un 50% más grande  
rectangle*0.5 //El rectángulo se hace un 50% más pequeño
```





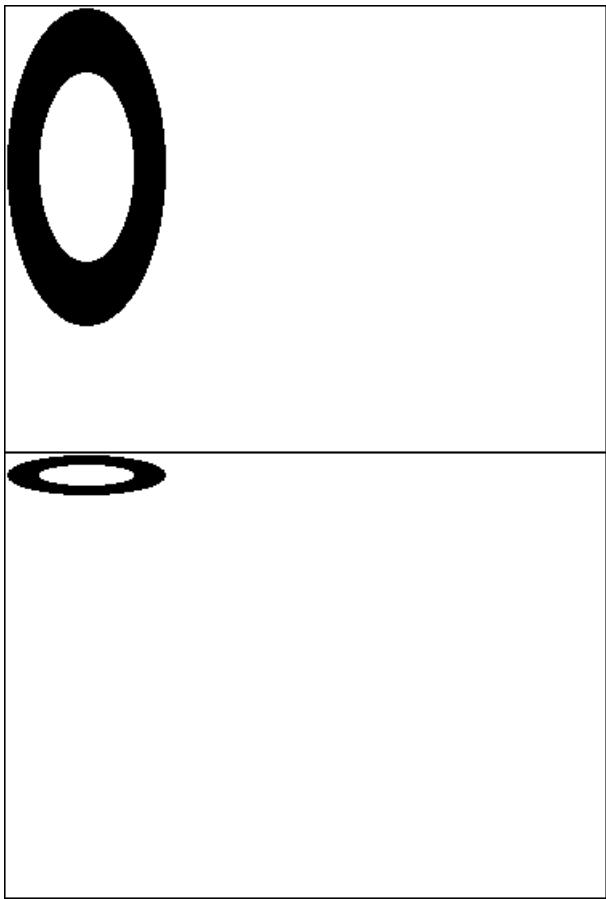
Escala horizontal

```
circle*+3 //El círculo se hace 3 veces más ancho  
circle*+0,25 //El ancho del círculo se convierte en una cuarta parte de lo que era
```



Escala vertical

```
circle*|2 //El círculo pasa a ser el doble de alto  
circle*|0.25 //La altura del círculo pasa a ser un cuarto de lo que era
```



# Puntero

Una variable o expresión puntero es una referencia a otra variable (incluyendo arrays y elementos de arrays), tabla, campo u objeto. No hay ningún campo de tipo Puntero.

Los punteros ofrecen una forma avanzada (en programación) de referirse a los datos. Cuando se utiliza el lenguaje, se accede a varios objetos -en particular, tablas, campos, variables, objetos y arrays- simplemente utilizando sus nombres. Sin embargo, con frecuencia es útil referirse a estos elementos y acceder a ellos sin conocer sus nombres. Esto es lo que los punteros le permiten hacer.

El concepto de los punteros no es tan raro en la vida cotidiana. A menudo se hace referencia a algo sin conocer su identidad exacta. Por ejemplo, puede decirle a un amigo: "Vamos a dar una vuelta en tu coche" en lugar de "Vamos a dar una vuelta en el coche con matrícula 123ABD." En este caso, se refiere al coche con matrícula 123ABD utilizando la frase "tu coche." La frase "coche con matrícula 123ABD" es como el nombre de un objeto, y usar la frase "tu coche" es como usar un puntero para referenciar el objeto.

Poder referirse a algo sin conocer su identidad exacta es muy útil. De hecho, su amigo podría comprarse un coche nuevo, y la frase "tu coche" seguiría siendo correcta: seguiría siendo un coche y usted podría seguir dando un paseo en él. Los punteros funcionan de la misma manera. Por ejemplo, un puntero podría referirse en un momento dado a un campo numérico llamado Edad, y más tarde referirse a una variable numérica llamada Vejez. En ambos casos, el puntero hace referencia a datos numéricos que podrían utilizarse en un cálculo.

Puede utilizar punteros para referenciar tablas, campos, variables, arrays, elementos de arrays y objetos. La siguiente tabla ofrece un ejemplo de cada tipo de datos:

Tipo	Referenciación	Uso	Asignación
Tabla	vpTable:=->[Table]	DEFAULT TABLE(vpTable->)	n/a
Campo	vpField:=->[Table]Field	ALERT(vpField->)	vpField->:="John"
Variable	vpVar:=->Variable	ALERT(vpVar->)	vpVar->:="John"
Array	vpArr:=->Array	SORT ARRAY(vpArr->;>)	COPY ARRAY (Arr;vpArr->)
array	vpElem:=->Array{1}	ALERT (vpElem->)	vpElem->:="John"
Objeto	vpObj:=->myObject	ALERT (vpObj->myProp)	vpObj->myProp:="John"

## Utilizar punteros: ejemplo básico

Lo más fácil es explicar el uso de los punteros mediante un ejemplo. Este ejemplo muestra cómo acceder a una variable a través de un puntero. Comenzamos creando una variable:

```
$MyVar:="Hello"
```

\$MyVar es ahora una variable que contiene la cadena "Hello." Ahora podemos crear un puntero a \$MyVar:

```
C_POINTER($MyPointer)  
$MyPointer:=->$MyVar
```

El símbolo `->` significa "obtener un puntero a." Este símbolo está formado por un guion seguido de un signo "mayor que". En este caso, obtiene el puntero que hace referencia o "apunta" a \$MyVar. Este puntero se asigna a MyPointer con el operador de asignación.

\$MyPointer es ahora una variable que contiene un puntero a \$MyVar. \$MyPointer no contiene "Hello", que es el valor en \$MyVar, pero se puede utilizar \$MyPointer para obtener este valor. La siguiente expresión devuelve el valor de \$MyVar:

```
$MyPointer->
```

En este caso, devuelve la cadena "Hello". El símbolo `->`, cuando sigue a un puntero, hace referencia al objeto apuntado. Esto se llama desreferenciación.

Es importante entender que se puede utilizar un puntero seguido del símbolo `->` en cualquier lugar donde se podría haber utilizado el objeto al que apunta el puntero. Esto significa que podría utilizar la expresión `$MyPointer->` en cualquier lugar en el que pudiera utilizar la variable original `$MyVar`. Por ejemplo, la siguiente línea muestra un cuadro de alerta con la palabra Hello:

```
ALERT($MyPointer->)
```

También puede utilizar `$MyPointer` para cambiar los datos en `$MyVar`. Por ejemplo, la siguiente instrucción almacena la cadena "Goodbye" en la variable `$MyVar`:

```
$MyPointer->:="Goodbye"
```

Si examina los dos usos de la expresión `$MyPointer->`, verá que actúa igual que si hubiera utilizado `$MyVar` en su lugar. En resumen, las dos líneas siguientes realizan la misma acción: ambas muestran un cuadro de alerta con el valor actual de la variable `$MyVar`:

```
ALERT($MyPointer->)
ALERT($MyVar)
```

Las siguientes dos líneas realizan la misma acción - ambas asignan la cadena "Goodbye" a `$MyVar`:

```
$MyPointer->:="Goodbye"
$MyVar:="Goodbye"
```

## Operadores en punteros

Con:

```
` vPtrA y vPtrB apuntan al mismo objeto
vPtrA:=>anObject
vPtrB:=>anObject
` vPtrC apunta a otro objeto
vPtrC:=>anotherObject
```

Operación	Sintaxis	Devuelve	Expresión	Valor
Igual	Puntero = Puntero	Booleano	vPtrA = vPtrB	True
			vPtrA = vPtrC	False
Desigualdad	Puntero # Puntero	Booleano	vPtrA # vPtrC	True
			vPtrA # vPtrB	False

## Principales usos

### Punteros a tablas

En cualquier lugar en el que el lenguaje espere ver una tabla, se puede utilizar un puntero desreferenciado a la tabla. Se crea un puntero a una tabla utilizando una línea de instrucción como esta:

```
$TablePtr:=->[anyTable]
```

También puede obtener un puntero a una tabla utilizando el comando `Table`:

```
$TablePtr:=Table(20)
```

Puedes utilizar el puntero desreferenciado en los comandos, así:

```
DEFAULT TABLE($TablePtr->)
```

## Punteros a campos

En cualquier lugar en el que el lenguaje espere ver un campo, se puede utilizar un puntero desreferenciado para referenciar el campo. Se crea un puntero a un campo utilizando una línea de instrucción como esta:

```
$FieldPtr:=->[aTable]ThisField
```

También puede obtener un puntero a un campo utilizando el comando `Campo`, por ejemplo:

```
$FieldPtr:=Field(1;2)
```

Puedes utilizar el puntero desreferenciado en los comandos, así:

```
OBJECT SET FONT($FieldPtr->;"Arial")
```

## Punteros a variables locales

Cuando se utilizan punteros a variables de proceso o locales, hay que asegurarse de que la variable a la que se apunta ya está definida cuando se utilice el puntero. Tenga en cuenta que las variables locales se borran cuando el método que las creó ha terminado su ejecución y las variables de proceso se borran al final del proceso que las creó. Cuando un puntero llama a una variable que ya no existe, esto provoca un error de sintaxis en modo interpretado (variable no definida) pero puede generar un error más grave en modo compilado.

Los punteros a variables locales permiten guardar las variables del proceso en muchos casos. Los punteros a variables locales sólo pueden utilizarse dentro del mismo proceso. En el depurador, cuando se muestra un puntero a una variable local que ha sido declarada en otro método, el nombre del método original se indica entre paréntesis, después del puntero. Por ejemplo, si se escribe en Method1:

```
$MyVar:="Hello world"  
Method2(->$MyVar)
```

En Method2, el depurador mostrará \$1 de la siguiente manera:

\$1	->\$MyVar (Method1)
-----	---------------------

El valor de 1 dólar será:

\$MyVar (Method1)	"Hello world"
-------------------	---------------

## Punteros a elementos del array

Puede crear un puntero a un elemento del array. Por ejemplo, las siguientes líneas crean un array y asignan un puntero al primer elemento del array a una variable llamada \$ElemPtr:

```
ARRAY REAL($anArray;10) //Crear un array  
$ElemPtr:==>$anArray{1} //Crear un puntero al elemento de array
```

Puede utilizar el puntero desreferenciado para asignar un valor al elemento, así:

```
$ElemPtr->:=8
```

## Punteros a arrays

Puede crear un puntero a un array. Por ejemplo, las siguientes líneas crean un array y asignan un puntero al array a una variable llamada \$ArrPtr:

```
ARRAY REAL($anArray;10) //Crear un array  
$ArrPtr:==>$anArray //Crear un puntero al array
```

Es importante entender que el puntero apunta al array; no apunta a un elemento del array. Por ejemplo, puede utilizar el puntero desreferenciado de las líneas anteriores de esta manera:

```
SORT ARRAY($ArrPtr->:>) //Ordenar el array
```

Si debe referirse al cuarto elemento del array utilizando el puntero, haga lo siguiente:

```
ArrPtr->{4}:=84
```

## Punteros como parámetros a los métodos

Puede pasar un puntero como parámetro de un método. Dentro del método, puede modificar el objeto referenciado por el puntero. Por ejemplo, el siguiente método, `takeTwo`, toma dos parámetros que son punteros. Cambia el objeto referenciado por el primer parámetro a caracteres en mayúsculas, y el objeto referenciado por el segundo parámetro a caracteres en minúsculas. Este es el método del proyecto:

```
//Método proyecto takeTwo  
//$1 – Puntero a un campo o variable de tipo cadena. Cambia la cadena a mayúsculas.  
//$2 – Puntero a un campo o variable de tipo cadena. Cambia la cadena a minúsculas.  
$1->:=Uppercase($1->)  
$2->:=Lowercase($2->)
```

La siguiente línea utiliza el método `takeTwo` para cambiar un campo a mayúsculas y para cambiar una variable a minúsculas:

```
takeTwo(>[myTable]myField;=>$MyVar)
```

Si el campo [myTable]myField contenía la cadena "jones", se cambiaría por la cadena "JONES". Si la variable \$MyVar contenía la cadena "HELLO", se cambiaría por la cadena "hello".

En el método `takeTwo`, y de hecho, siempre que se utilicen punteros, es importante que el tipo de datos del objeto al que se hace referencia sea correcto. En el ejemplo anterior, los punteros deben apuntar a algo que contenga una

cadena o texto.

## Punteros a punteros

Si realmente le gusta complicar las cosas, puede utilizar punteros para referenciar otros punteros. Considere este ejemplo:

```
$MyVar:="Hello"  
$PointerOne:==>$MyVar  
$PointerTwo:==>$PointerOne  
($PointerTwo->)->:="Goodbye"  
ALERT(($PointerTwo->)->)
```

Muestra un cuadro de alerta con la palabra "Goodbye".

A continuación se explica cada línea del ejemplo:

- \$MyVar:="Hello" --> Esta línea pone la cadena "Hello" en la variable \$MyVar.
- \$PointerOne:==>\$MyVar --> \$PointerOne ahora contiene un puntero a \$MyVar.
- \$PointerTwo:==>\$PointerOne --> \$PointerTwo (una nueva variable) contiene un puntero a \$PointerOne, que a su vez apunta a \$MyVar.
- (\$PointerTwo->)->:="Goodbye" --> \$PointerTwo-> referencia el contenido de \$PointerOne, que a su vez referencia \$MyVar. Por lo tanto, (\$PointerTwo->)-> referencia el contenido de \$MyVar. Así que en este caso, a \$MyVar se le asigna "Goodbye".
- ALERT (( \$PointerTwo->)->) --> Lo mismo que: \$PointerTwo-> referencia el contenido de \$PointerOne, que a su vez referencia \$MyVar. Por lo tanto, (\$PointerTwo->)-> referencia el contenido de \$MyVar. Por lo tanto, (\$PointerTwo->)-> referencia el contenido de \$MyVar.

La siguiente línea pone "Hello" en \$MyVar:

```
($PointerTwo->)->:="Hello"
```

La siguiente línea obtiene "Hello de \$MyVar y lo pone en \$NewVar:

```
$NewVar:=($PointerTwo->)->
```

Importante: la desreferenciación múltiple requiere paréntesis.

# Cadena

Cadena es un término genérico utilizado para:

- Los campos o variables de tipo texto: un campo, variable o expresión de tipo texto puede contener de 0 a 2 GB de texto.
- Campos alfanuméricos: un campo alfanumérico puede contener de 0 a 255 caracteres (límite fijado al definir el campo).

## Constantes literales de tipo cadena

Una constante literal de tipo cadena se encierra entre comillas dobles y rectas ("..."). Estos son algunos ejemplos:

```
"Añadir registros"  
"No se han encontrado registros."  
"Factura"
```

Una cadena vacía se especifica con dos comillas sin nada entre ellas ("").

## Secuencias de escape

Las siguientes secuencias de escape pueden utilizarse dentro de las cadenas:

Secuencias de escape	Carácter reemplazado
¥n	LF (Retorno línea)
¥t	HT (Tabulación)
¥r	CR (Retorno carro)
¥¥	¥ (Barra invertida)
¥"	" (Comillas)

Nota: el carácter ¥ (barra invertida) se utiliza como separador en las rutas de acceso en Windows. Por lo tanto, debe utilizar una doble barra invertida ¥¥ en las rutas cuando quiera tener una barra invertida delante de un carácter utilizado en una de las secuencias de escape reconocidas por 4D (por ejemplo, "C:¥MisDocumentos¥¥Nuevos.txt").

## Operadores de cadenas

Operación	Sintaxis	Devuelve	Expresión	Valor
Concatenación	Cadena + Cadena	Cadena	"abc" + "def"	"abcdef"
Repetición	Cadena * Número	Cadena	"ab" * 3	"ababab"
Igual	Cadena = Cadena	Booleano	"abc" = "abc"	True
			"abc" = "abd"	False
Desigualdad	Cadena # Cadena	Booleano	"abc" # "abd"	True
			"abc" # "abc"	False
Mayor que	Cadena > Cadena	Booleano	"abd" > "abc"	True
			"abc" > "abc"	False
Menor que	Cadena < Cadena	Booleano	"abc" < "abd"	True
			"abc" < "abc"	False
Mayor o igual que	Cadena >= Cadena	Booleano	"abd" >= "abc"	True
			"abc" >= "abd"	False
Menor o igual que	Cadena <= Cadena	Booleano	"abc" <= "abd"	True
			"abd" <= "abc"	False
Contiene palabra clave	Cadena % Cadena	Booleano	"Alpha Bravo" % "Bravo"	True
			"Alpha Bravo" % "ravo"	False
	Imagen % Cadena	Booleano	Picture_expr % "Mer"	True (*)

(\*) Si la palabra clave "Mer" está asociada a la imagen almacenada en la expresión imagen (campo o variable).

## Comparaciones de cadenas

- Las cadenas se comparan carácter por carácter (excepto en el caso de la búsqueda por [palabras clave](#), ver más abajo).
- Cuando se comparan cadenas, se ignoran las mayúsculas y minúsculas de los caracteres; así, "a"="A" devuelve `TRUE`. Para saber si los caracteres están en mayúsculas o minúsculas, compare sus códigos de caracteres. Por ejemplo, la siguiente expresión devuelve `FALSE` :

```
Character code("A")=Character code("a") // porque 65 no es igual a 97
```

- Cuando se comparan las cadenas, se tienen en cuenta los caracteres diacríticos. Por ejemplo, las siguientes expresiones devuelven `TRUE` :

```
"ñ"="ñ"  
"ñ"="Ñ"  
"A"="å"  
// etc
```

Nota: la comparación de cadenas tiene en cuenta las especificidades del lenguaje definido para el archivo de datos 4D (que no siempre es el mismo que el lenguaje definido para el sistema).

## Arroba (@)

El lenguaje 4D soporta @ como carácter comodín. Este carácter se puede utilizar en cualquier comparación de cadenas para que coincida con cualquier número de caracteres. Por ejemplo, la siguiente expresión es `TRUE` :

```
"abcdefgij"="abc@"
```

El carácter comodín debe utilizarse dentro del segundo operando (la cadena de la derecha) para que coincida con cualquier número de caracteres. La siguiente expresión es `FALSE`, porque la `@` se considera sólo como un carácter en el primer operando:

```
"abc@"="abcdefgij"
```

El comodín significa "uno o más caracteres o nada". Las expresiones siguientes son `TRUE`:

```
"abcdefgij"="abcdefgij@"
"abcdefgij"="@abcdefgij"
"abcdefgij"="abcd@efgij"
"abcdefgij"="@abcdefgij@"
"abcdefgij"="@abcde@fghij@"
```

Por otro lado, sea cual sea el caso, una comparación de cadenas con dos comodines consecutivos siempre devolverá `FALSE`. La siguiente expresión es `FALSE`:

```
"abcdefgij"="abc@@fg"
```

Cuando el operador de comparación es `o` contiene un símbolo `<` o `>`, sólo se soporta la comparación con un único comodín situado al final del operando:

```
"abcd"<="abc@" // Comparación válida
"abcd"<="abc@ef" //No es una comparación válida
```

Si desea ejecutar comparaciones o consultas utilizando `@` como carácter (y no como comodín), debe utilizar la instrucción `Character code(At sign)`. Imagine, por ejemplo, que quiere saber si una cadena termina con el carácter `@`. La siguiente expresión (si `$vsValue` no está vacío) es siempre `TRUE`:

```
($vsValue[[Length($vsValue)]]="@")
```

La siguiente expresión se evaluará correctamente:

```
(Character code($vsValue[[Length($vsValue)]]))#64)
```

Nota: una opción 4D del modo Diseño permite definir cómo se interpreta el carácter `@` cuando se incluye en una cadena de caracteres.

## Palabras claves

A diferencia de otras comparaciones de cadenas, la búsqueda por palabras clave busca "palabras" en los "textos": las palabras se consideran tanto individualmente como en su conjunto. El operador `%` siempre devuelve `False` si la consulta se refiere a varias palabras o sólo a una parte de ellas (por ejemplo, una sílaba). Las "palabras" son cadenas de caracteres rodeadas de "separadores", que son espacios y caracteres de puntuación y guiones. Un apóstrofe, como en "Today's", se considera normalmente como parte de la palabra, pero se ignorará en ciertos casos (ver las reglas más abajo). Los números se pueden buscar porque se evalúan como un todo (incluidos los símbolos decimales). Los demás símbolos (moneda, temperatura, etc.) serán ignorados.

```

"Alpha Bravo Charlie"%"Bravo" // DevuelveTrue
"Alpha Bravo Charlie"%"vo" // Devuelve False
"Alpha Bravo Charlie"%"Alpha Bravo" // Devuelve False
"Alpha,Bravo,Charlie"%"Alpha" // Devuelve True
"Software and Computers"%"comput@" // Devuelve True

```

Notas: - 4D utiliza la librería ICU para comparar cadenas (utilizando operadores <>=#) y detectar palabras claves. Para más información sobre las normas aplicadas, consulte la siguiente dirección:  
[http://www.unicode.org/reports/tr29/#Word\\_Boundaries](http://www.unicode.org/reports/tr29/#Word_Boundaries). - En la versión japonesa, en lugar de ICU, 4D utiliza por defecto Mecab para detectar las palabras claves.

## Símbolos de indice de cadena

Los símbolos de indice de cadena son: [...]

Estos símbolos se utilizan para referirse a un solo carácter dentro de una cadena. Esta sintaxis permite direccionar individualmente los caracteres en un campo o una variable de tipo Alfa o Texto.

Si los símbolos de referencia de caracteres aparecen a la izquierda del operador de asignación (:=), se asigna un carácter a la posición referenciada en la cadena. Por ejemplo, si vsName no es una cadena vacía, la siguiente línea pone el primer carácter de vsName en mayúsculas:

```

If(vsNom#"")

    vsNom[[1]]:=Uppercase(vsNom[[1]])

End if

```

En caso contrario, si los símbolos de referencia de caracteres aparecen dentro de una expresión, devuelven el carácter (al que se refieren) como una cadena de 1 carácter. Por ejemplo:

```

//El siguiente ejemplo comprueba si el último carácter de vtText es una arroba "@". If(vtText#"")
    If(Character code(Substring(vtText;Length(vtText);1))=At sign)
    //...
    End if
End if

//Utilizando la sintaxis de los caracteres de indice, se escribiría de forma más sencilla:
If(vtText#"")
    If(Character code(vtText[[Length(vtText)]])=At sign)
    // ...
    End if
End if

```

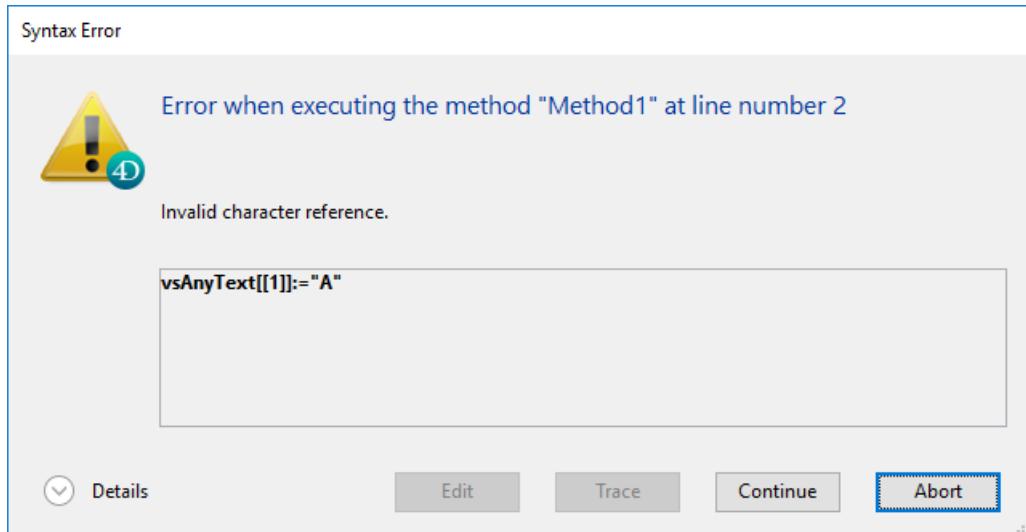
## Nota avanzada sobre la referencia de caracteres inválidos

Al utilizar los símbolos de indice de cadena, debe dirigirse a los caracteres existentes en la cadena de la misma manera que se dirige a los elementos existentes de un array. Por ejemplo, si se referencia el 20º carácter de una variable cadena, esta variable DEBE contener al menos 20 caracteres.

- No hacerlo, en modo interpretado, no causa un error de sintaxis.
- No hacerlo, en modo compilado (sin opciones), puede llevar a la corrupción de la memoria, si, por ejemplo, se escribe un carácter más allá del final de una cadena o un texto.
- Si no lo hace, en el modo compilado, se produce un error con la opción Range Checking On. Por ejemplo, ejecutando el siguiente código:

```
//Muy malo y desagradable, iBoo!
//Muy malo y desagradable, iBoo!
//Muy malo y desagradable, iBoo!
vsAnyText:=""
vsAnyText[[1]]:="A"
```

provocará el error de tiempo de ejecución que se muestra aquí:



## Ejemplo

El siguiente método proyecto pone en mayúsculas el primer carácter de cada palabra del texto recibido como parámetro y devuelve el texto resultante en mayúsculas:

```
//Método proyecto Capitalize_text
//Capitalize_text ( Texto ) -> Texto
//Capitalize_text ( Texto fuente ) -> Texto en mayúsculas

$0:=$1
$vLen:=Length($0)
If($vLen>0)
    $0[[1]]:=Uppercase($0[[1]])
    For($vlChar;1;$vLen-1)
        If(Position($0[[vlChar]]; " !&()-{}:;<>?/,.=+*")>0)
            $0[[vlChar+1]]:=Uppercase($0[[vlChar+1]])
        End if
    End for
End if
```

Por ejemplo, la línea:

```
ALERT(Capitalize_text("hola, me llamo jane doe y me presento a la presidencia"))
```

muestra la alerta que aparece aquí:

Alert



Hello, My Name Is Jane Doe And I'm Running For President!

OK

# Hora

Las variables, campos o expresiones de tipo Hora pueden estar comprendidas entre 00:00:00 y 596,000:00:00.

Las horas están en formato de 24 horas.

Un valor de tiempo puede ser tratado como un número. El número devuelto de una hora es el número de segundos desde la medianoche (00:00:00) que representa esa hora.

Nota: en el manual de *referencia del lenguaje 4D*, los parámetros de tipo Hora en las descripciones de los comandos se llaman Hora, excepto cuando se indique lo contrario.

## Constantes literales de tipo hora

Una constante hora está rodeada de signos de interrogación (?...?).

Una constante hora se ordena hora:minuto:segundo, con dos puntos (:) para separar cada parte. Las horas se especifican en formato de 24 horas.

Estos son algunos ejemplos de constantes de tipo hora:

```
?00:00:00? ?00:00:00? ` media noche
?09:30:00? ?00:00:00? ` media noche
?09:30:00? ` 9:30 am
?13:01:59? ?00:00:00? ` media noche
?09:30:00? ` 9:30 am
?13:01:59? ` 1 pm, 1 minuto, y 59 segundos ?00:00:00? ` media noche
?09:30:00? ` 9:30 am
?13:01:59? ` 1 pm, 1 minuto, y 59 segundos ?00:00:00? ` media noche
?09:30:00? ` 9:30 am
?13:01:59? ?00:00:00? ` media noche
?09:30:00? ` 9:30 am
?13:01:59? ` 1 pm, 1 minuto, y 59 segundos ?00:00:00? ` media noche
?09:30:00? ` 9:30 am
?13:01:59? ` 1 pm, 1 minuto, y 59 segundos
```

Una hora nula se escribe ?00:00:00?

Consejo: el Editor de métodos incluye un acceso directo para introducir una hora nula. Para escribir una hora nula, introduzca el carácter de interrogante (?) y pulse Intro.

## Operadores de horas

Operación	Sintaxis	Devuelve	Expresión	Valor
Adición	Hora + Hora	Hora	?02:03:04? ?02:03:04? + ?01:02:03?	?03:05:07?
Resta	Hora - Hora	Hora	?02:03:04? - ?01:02:03?	?01:01:01?
Adición	Hora + Número	Número	?02:03:04? ?02:03:04? + 65	7449
Resta	Hora - Número	Número	?02:03:04? - 65	7319
Multiplicación	Hora * Número	Número	?02:03:04? * 2	14768
División	Hora / Número	Número	?02:03:04? ?02:03:04? / 2	3692
División entera	Hora ¥ Número	Número	?02:03:04? ¥ 2	3692
Módulo	Hora % Hora	Hora	?20:10:00? ?20:10:00? % ?04:20:00? ?20:10:00? % ?04:20:00? ?20:10:00? % ?04:20:00?	?02:50:00?
Módulo	Hora % Número	Número	?02:03:04? ?02:03:04? % 2	0
Igual	Hora = Hora	Booleano	?01:02:03? ?01:02:03? = ?01:02:03?	True
			?01:02:03? ?01:02:03? = ?01:02:04?	False
Desigualdad	Hora # Hora	Booleano	?01:02:03? # ?01:02:04?	True
			?01:02:03? # ?01:02:03?	False
Mayor que	Hora > Hora	Booleano	?01:02:03? # ?01:02:04? ?01:02:03? ?01:02:04? > ?01:02:03?	True
			?01:02:03? ?01:02:03? ?01:02:04? > ?01:02:03?	False
Menor que	Hora < Hora	Booleano	?01:02:03? ?01:02:03? < ?01:02:04?	True
			?01:02:03? ?01:02:03? ?01:02:03? < ?01:02:04?	False
Mayor o igual que	Hora >= Hora	Booleano	?01:02:03? ?01:02:03? ?01:02:03? >=?01:02:03?	True
			?01:02:03? ?01:02:04? > ?01:02:03?	False
Menor o igual que	Hora <= Hora	Booleano	?01:02:03? ?01:02:03? <=?01:02:03?	True
			?01:02:03? # ?01:02:04? ?01:02:03? <=?01:02:03?	False

## Ejemplo 1

Para obtener una expresión de tipo hora a partir de una expresión que combina una expresión de hora con un número, utilice los comandos `Time` y `Time string`.

Puede combinar expresiones de los tipos hora y número utilizando las funciones `Time` o `Current time`:

```
//La siguiente línea asigna a $vlSeconds el número de segundos  
//que transcurrirán entre la medianoche y una hora a partir de ahora  
$vlSeconds:=Current time+3600  
//La siguiente línea asigna a $vhSoon la hora que será dentro de una hora  
$vhSoon:=Time(Current time+3600)
```

La segunda línea podría escribirse de forma más sencilla:

```
// La siguiente línea asigna a $vhSoon la hora que será en una hora  
$vhSoon:=Current time+?01:00:00?
```

## Ejemplo 2

El operador Modulo puede utilizarse, más concretamente, para sumar tiempos que tengan en cuenta el formato de 24 horas:

```
$t1:=?23:00:00? // Son las 23:00 p.m. // Son las 23:00 horas  
// Queremos añadir 2 horas y media  
$t2:=$t1 +?02:30:00? // Con una simple adición, $t2 es ?25:30:00?  
$t2:=($t1 +?02:30:00?)%?24:00:00? // $t2 es ?01:30:00? y es la 1:30 de la mañana siguiente
```

# Variant

Variant es un tipo de variable que permite encapsular datos de todo tipo válido y estándar en una variable. Normalmente, este tipo de variable puede utilizarse para escribir código genérico que devuelva o reciba valores cuyo tipo no se conoce. Este es el caso, por ejemplo, del código que maneja los atributos de objeto.

Una variable de tipo variant puede contener un valor de los tipos de datos siguientes:

- BLOB
- booleano
- colección
- fecha
- entero largo
- objeto
- imagen
- puntero
- real
- texto
- time
- null
- indefinido

Las matrices no pueden almacenarse en variables de tipo variant.

Tanto en modo interpretado como en compilado, a una misma variable variant se le pueden asignar contenidos de diferentes tipos. A diferencia de los tipos de variable estándar, el tipo de contenido de la variable variant es diferente del tipo de variable variant mismo. Por ejemplo:

```
C_VARIANT($variant)

$variant:="hello world"
$vtype:=Type($variant) // 12 (Is variant)
$value:=Value type($variant) // 2 (Is text)

$variant:=42
$vtype:=Type($variant) // 12 (Is variant)
$value:=Value type($variant) // 1 (Is real)
```

Se pueden utilizar variables variant en cualquier lugar donde se esperen variables, sólo hay que asegurarse de que el tipo de datos del contenido de la variable es del tipo esperado. Al acceder a las variables de las variantes, sólo se tiene en cuenta su valor actual. Por ejemplo:

```
C_VARIANT($v)
$v:="hello world"
$v2:=$v //asignar una variable a otra variable

$t:=Type($v) // 12 (Is variant)
$t2:=Type($v2) // 2 (Is text)
```

Variant se puede utilizar para declarar parámetros de métodos (\$0, \$1,...) que pueden ser de varios tipos. En este caso, puede construir su código probando el tipo de valor del parámetro, por ejemplo:

```
C_VARIANT($1)
Case of
: (Value type($1)=Is longint)
...
: (Value type($1)=Is text)
...
End case
```

Cuando las variables variant no son necesarias (es decir, cuando se conoce el tipo de datos), se recomienda utilizar variables de tipo estándar. Las variables de tipo estándar ofrecen un mejor rendimiento, hacen que el código sea más claro y son útiles para que el compilador evite los errores relacionados con el paso de tipos de datos inesperados.

# Variables

Los datos en 4D se almacenan de dos formas fundamentalmente diferentes. Los campos almacenan los datos permanentemente en el disco; las variables almacenan los datos en la memoria de forma temporal.

Cuando define su base, especifique a 4D los nombres y los tipos de campos que desea utilizar. Las variables son muy parecidas, también se les da nombres y tipos diferentes (ver [Tipos de datos](#)).

Una vez creada, puede utilizar una variable dondequiero que la necesite en su aplicación. Por ejemplo, podría necesitar almacenar una variable texto en un campo del mismo tipo:

```
[MyTable]MyField:=MyText
```

Las variables son objetos del lenguaje; puede crear y utilizar variables que nunca aparecerán en la pantalla. En sus formularios, puede mostrar variables en la pantalla (excepto de los punteros y de los BLOB), introducir datos en ellas e imprimirlas en informes. De este modo, las variables de área introducibles y no introducibles actúan igual que los campos, y los mismos controles integrados están disponibles al crearlos. Las variables de formulario también pueden controlar botones, list boxes, áreas de desplazamiento, botones imagen, etc., o mostrar los resultados de cálculos que no necesitan ser guardados.

## Declaración de variables

Las variables se crean declarándolas. El lenguaje 4D ofrece dos formas de declarar las variables:

- utilizando la palabra clave `var` (recomendado, especialmente si su código utiliza objetos y clases),
- utilizando uno de los comandos del lenguaje 4D del tema "Compilador" o "Arrays" (lenguaje clásico únicamente).

Note: aunque no se suele recomendar, puede crear variables básicas simplemente utilizándolas; no es necesario definirlas formalmente. Por ejemplo, si declara una variable que contenga la fecha actual más 30 días, puede escribir:

```
MyDate:=Current date+30 //MyDate is created
// 4D guesses it is of date type
// and assigns the current date plus 30 days
```

### Utilizando la palabra clave `var`

Se recomienda declarar las variables utilizando la palabra clave `var` ya que esta sintaxis permite vincular las variables objeto con las clases. Using this syntax enhances code editor suggestions and type-ahead features.

Para declarar una variable de cualquier tipo con la palabra clave `var`, utilice la siguiente sintaxis:

```
var <varName>{; <varName2>;...}{ : <varType>}
```

Por ejemplo:

```
var $myText : Text //una variable texto
var myDate1; myDate2 : Date //varias variables fecha
var $myFile : 4D.File //una variable objeto clase archivo
var $myVar //una variable variant
```

`varName` es el nombre de la variable, debe cumplir con las [reglas 4D](#) sobre identificadores.

Esta sintaxis sólo admite declaraciones de [variables locales y proceso](#), por lo que se excluyen las [variables interprocesos](#) y los [arrays](#).

`varType` puede ser:

- un [tipo básico](#), en cuyo caso la variable contiene un valor del tipo declarado,
- una [referencia de clase](#) (clase 4D o clase usuario), en cuyo caso la variable contiene una referencia a un objeto de la clase definida.

Si `varType` se omite, una variable de tipo variant se crea.

La siguiente tabla enumera todos los valores `varType` soportados:

varType	Contenido
Texto	Valor texto
Fecha	Valor fecha
Hora	Valor Hora
Booleano	Valor booleano
Integer	Valor entero largo
Real	Valor real
Puntero	Valor puntero
Imagen	Valor imagen
Blob	Valeor Blob escalar
Collection	Valor colección
Variant	Valor variant
Objeto	Objeto con clase por defecto (4D.object)
4D.<className>	Objeto del nombre de la clase 4D
cs.<className>	Objeto del nombre de la clase usuario

## Ejemplos

- Para declarar variables básicas locales y de proceso:

```
var $myText; myText; $vt : Text
var myVar //variant

var $o : Object
//equivalente a:
var $o : 4D.Object
//también equivalente a C_OBJECT($o)
```

- Para declarar las variables objeto de la clase 4D:

```
var $myFolder : 4D.Folder
var $myFile : 4D.File
```

- Para declarar las variables objeto de la clase usuario:

```
var $myClass : cs.MyClass
var $dataclass : cs.Employee
var $entity : cs.EmployeeEntity
```

## Utilizando un C\_ directive

Nota: los parámetros \$1, \$2... pasados a los métodos son variables locales. Se recomienda utilizar la palabra clave **var**.

Las directivas del tema "Compilador" permiten declarar variables de tipos básicos.

Por ejemplo, si se quiere definir una variable de tipo texto, se escribe:

```
C_TEXT(myText)
```

A continuación se presentan algunas declaraciones de variables básicas:

```
C_BLOB(vxMyBlob) // La variable proceso vxMyBlob se declara como una variable de tipo BLOB  
C_DATE($vdCurDate) // La variable local $vdCurDate se declara como una variable de tipo Fecha  
C_LONGINT(vg1;vg2;vg3) // Las 3 variables de proceso vg1, vg2 y vg3 se declaran como variables de tipo LONGINT  
C_OBJECT($vObj) // La variable local $vObj se declara como una variable de tipo Objeto  
C_COLLECTION($vCol) // La variable local $vCol se declara como una variable de tipo Colección
```

Nota: los array son un tipo particular de variables (un array es una serie ordenada de variables del mismo tipo). Los arrays se declaran con comandos específicos, como `ARRAY LONGINT(alAnArray;10)`. Para más información, consulte [Arrays](#).

## Asignar los valores

Los datos pueden introducirse y copiarse en variables y arrays. Poner datos en una variable se llama **asignar los datos a la variable** y se hace con el operador de asignación (`:=`). El operador de asignación también se utiliza para asignar datos a campos.

El operador de asignación es un primer medio para crear una variable e introducir datos en ella. Se escribe el nombre de la variable que se quiere crear a la izquierda del operador de asignación. Por ejemplo:

```
MyNumber:=3
```

crea la variable *MyNumber* y pone en ella el número 3. Si *MyNumber* ya existe, entonces toma el valor 3.

Normalmente no se recomienda crear variables sin [declarar su tipo](#).

Por supuesto, las variables no serían muy útiles si no se pudieran obtener valores de ellas. Una vez más, se utiliza el operador de asignación. Si necesita poner el valor de *MyNumber* en un campo llamado *[Products]Size*, escribiría *MyNumber* a la derecha del operador de asignación:

```
[Products]Size:=MyNumber
```

En este caso, *[Products]Size* sería igual a 3. Este ejemplo es bastante sencillo, pero ilustra la forma fundamental en que se transfieren los datos de un lugar a otro utilizando el lenguaje.

Los valores se asignan a los elementos del array utilizando llaves (`{...}`):

```
atNames{1}:="Richard"
```

## Variables locales, proceso e interproceso

Puedes crear tres tipos de variables: local, proceso, e interproceso. La diferencia entre los tres tipos de elementos es su

alcance, o los objetos para los que están disponibles.

## Variables locales

Una variable local, como su nombre indica, es local a un método, accesible sólo dentro del método en el que fue creada y no accesible fuera de ese método. Ser local a un método se conoce formalmente como ser de "alcance local." Las variables locales se utilizan para restringir una variable para que funcione sólo dentro del método.

Es posible que desee utilizar una variable local para:

- Evitar conflictos con los nombres de otras variables
- Utilizar los datos temporalmente
- Reducir el número de variables proceso

El nombre de una variable local siempre comienza por el signo dólar (\$) y puede contener hasta 31 caracteres adicionales. Si introduce un nombre más largo, 4D lo trunca a la longitud adecuada.

Cuando se trabaja en un proyecto de aplicación con muchos métodos y variables, a menudo se encuentra que se necesita utilizar una variable sólo dentro del método en el que se está trabajando. Puede crear y utilizar una variable local en el método sin preocuparse de si ha utilizado el mismo nombre de variable en otro lugar.

Con frecuencia, en una aplicación, se necesitan pequeñas piezas de información del usuario. El comando `Request` puede obtener esta información. Muestra una caja de diálogo con un mensaje que solicita al usuario una respuesta. Cuando el usuario introduce la respuesta, el comando devuelve la información que el usuario introdujo. Generalmente no es necesario mantener esta información en sus métodos durante mucho tiempo. Esta es una forma típica de utilizar una variable local. Aquí un ejemplo:

```
$vsID:=Request("Introduzca su identificación:")
If(OK=1)
    QUERY( [People];[People] ID =$vsID)
End if
```

Este método simplemente pide al usuario que introduzca un ID. Pone la respuesta en una variable local, \$vsID, y luego busca el ID que el usuario introdujo. Cuando este método termina, la variable local \$vsID se borra de la memoria. Este funcionamiento está bien, porque la variable se necesita sólo una vez y sólo en este método.

Nota de compatibilidad: no se recomienda esta funcionalidad para declarar variables dentro de métodos. Se recomienda utilizar la palabra clave `var`.

## Variables proceso

Una variable proceso sólo está disponible dentro de un proceso. Es accesible al método del proceso y a todos los métodos llamados desde el proceso.

Una variable proceso no tiene un prefijo antes de su nombre. Un nombre de variable proceso puede contener hasta 31 caracteres.

En modo interpretado, las variables se mantienen dinámicamente; se crean y se borran de la memoria "sobre la marcha". En modo compilado, todos los procesos que se crean (procesos usuario) comparten la misma definición de variables proceso, pero cada proceso tiene una instancia diferente para cada variable. Por ejemplo, la variable miVar es una variable en el proceso P\_1 y otra en el proceso P\_2.

Un proceso puede leer y escribir las variables proceso de otro proceso utilizando los comandos `GET PROCESS VARIABLE` y `SET PROCESS VARIABLE`. Es una buena práctica de programación restringir el uso de estos comandos a la situación para la que fueron creados en 4D:

- Comunicación interprocesos en lugares específicos de su código
- Gestión de arrastrar y soltar interproceso
- En Cliente/Servidor, la comunicación entre los procesos en las máquinas cliente y los procedimientos almacenados ejecutados en las máquinas servidoras

Para más información, consulte el capítulo Procesos y la descripción de estos comandos.

## Variables interproceso

Las variables interproceso están disponibles en todo el proyecto y son compartidas por todos los procesos cooperativos. Se utilizan principalmente para compartir información entre procesos.

No se recomienda el uso de variables interproceso, ya que no están disponibles para los procesos apropiativos y tienden a hacer que el código sea menos mantenible.

El nombre de una variable interproceso siempre comienza con los símbolos (<>) -un signo "menor que" seguido de un signo "mayor que"- seguido de 31 caracteres.

En modo cliente/servidor, cada máquina (cliente y servidor) comparten la misma definición de las variables interproceso, pero cada máquina tiene una instancia diferente para cada variable.

# Arrays

Un array es una serie ordenada de variables del mismo tipo. Cada variable se llama un elemento del array. Un array recibe su tamaño cuando se crea; luego se puede redimensionar tantas veces como sea necesario añadiendo, insertando o eliminando elementos, o redimensionando el array utilizando el mismo comando utilizado para crearlo. Los elementos del array se numeran de 1 a N, siendo N el tamaño del array. Un array siempre tiene un [elemento cero](#) especial. Los arrays son variables 4D. Como toda variable, un array tiene un alcance y sigue las reglas del lenguaje 4D, aunque con algunas diferencias únicas.

En la mayoría de los casos, se recomienda utilizar colecciones en lugar de arrays. Las colecciones son más flexibles y ofrecen una amplia gama de métodos dedicados. Para más información, consulte la sección [Colección](#).

## Crear arrays

Se crea un array con uno de los comandos de declaración del tema "Arrays". Cada comando de declaración de arrays puede crear o redimensionar arrays unidimensionales o bidimensionales. Para más información sobre los arrays bidimensionales, consulte la sección [arrays bidimensionales](#).

La siguiente línea de código crea (declara) un array de enteros de 10 elementos:

```
ARRAY INTEGER(aiAnArray;10)
```

A continuación, el siguiente código redimensiona ese mismo array a 20 elementos:

```
ARRAY INTEGER(aiAnArray;20)
```

A continuación, el siguiente código redimensiona ese mismo array a 0 elementos:

```
ARRAY INTEGER(aiAnArray;0)
```

## Asignar valores en arrays

Para referirse a los elementos de un array se utilizan llaves ({}). Dentro de las llaves se utiliza un número para dirigirse a un elemento concreto; este número se denomina número de elemento. Las siguientes líneas ponen cinco nombres en el array llamado atNames y luego los muestran en ventanas de alerta:

```
ARRAY TEXT(atNames;5)
atNames{1}:="Richard"
atNames{2}:="Sarah"
atNames{3}:="Sam"
atNames{4}:="Jane"
atNames{5}:="John"
For($vlElem;1;5)
    ALERT("The element #"+String($vlElem)+" is equal to: "+atNames{$vlElem})
End for
```

Tenga en cuenta la sintaxis atNames{\$vlElem}. En lugar de especificar un literal numérico como atNames{3}, puede utilizar una variable numérica para indicar a qué elemento de un array se dirige. Utilizando la iteración que ofrece una estructura de bucle ( `For...End for`, `Repeat...Until` o `While...End while` ), las piezas compactas de código pueden dirigirse a todos o a parte de los elementos de un array.

Importante: tenga cuidado de no confundir el operador de asignación (:=) con el operador de comparación, igual (=). La asignación y la comparación son operaciones muy diferentes.

## Asignación de un array a otro array

A diferencia de las variables de tipo texto o cadena, no se puede asignar un array a otro. Para copiar (asignar) un array a otro, utilice `COPY ARRAY`.

## Utilizar el elemento cero de un array

Un array siempre tiene un elemento cero. Aunque el elemento cero no se muestra cuando un array soporta un objeto de formulario, no hay ninguna restricción(\*) para utilizarlo en el lenguaje.

He aquí otro ejemplo: quiere inicializar un objeto de formulario con un valor texto pero sin definir un valor por defecto. Puede utilizar el elemento cero del array:

```
// método para un combo box o una lista desplegable
// vinculado al array de variables atName
Case of
  :(Form event code=On Load)
    // Inicializar el array (como se muestra más arriba)
    // Pero utiliza el elemento cero
    ARRAY TEXT(atName;5)
    atName{0}:=Seleccione un elemento"
    atName{1}:="Texto1"
    atName{2}:="Texto2"
    atName{3}:="Texto3"
    atName{4}:="Texto4"
    atName{5}:="Texto5"
    // Posicionar el array en el elemento 0
    atName:=0
End case
```

(\*) Sin embargo, hay una excepción: en un array tipo List Box, el elemento cero se utiliza internamente para almacenar el valor anterior de un elemento que se está editando, por lo que no es posible utilizarlo en este contexto particular.

## Arrays de dos dimensiones

Cada comando de declaración de arrays puede crear o redimensionar arrays unidimensionales o bidimensionales. Ejemplo:

```
ARRAY TEXT(atTopics;100;50) // Crear un array texto compuesto por 100 líneas de 50 columnas
```

Los arrays de dos dimensiones son esencialmente objetos de lenguaje; no se pueden mostrar ni imprimir.

En el ejemplo anterior:

- `atTopics` es una array de dos dimensiones
- `atTopics{8}{5}` es el 5º elemento (5ª columna...) de la 8ª fila
- `atTopics{20}` es la vigésima fila y es a su vez un array de una dimensión
- `Tamaño del array(atTopics)` devuelve 100, que es el número de filas
- `Tamaño de array(atTopics{17})` devuelve 50, que es el número de columnas de la línea 17

En el siguiente ejemplo, un puntero a cada campo de cada tabla de la base se almacena en un array de dos dimensiones:

```

C_LONGINT($vlLastTable;$vlLastField)
C_LONGINT($vlFieldNumber)
// Crear tantas líneas (vacías y sin columnas) como tablas haya
$vlLastTable:=Get last table number
ARRAY POINTER(<>apFields;$vlLastTable;0) /Array 2D con X líneas y cero columnas
// Para cada tabla
For($vlTable;1;$vlLastTable)
    If(Is table number valid($vlTable))
        $vlLastField:=Obtener el número del último campo($vlTable)
    // Dar valor a los elementos
    $vlColumnNumber:=0
    For($vlField;1;$vlLastField)
        If(Is field number valid($vlTable;$vlField))
            $vlColumnNumber:=$vlColumnNumber+1
    //Inserta una columna en una línea de la tabla en curso
    INSERT IN ARRAY(<>apFields{$vlTable};$vlColumnNumber;1)
    //Asignar la "celda" con el puntero
    <>apFields{$vlTable}{$vlColumnNumber}:=Field($vlTable;$vlField)
    End if
    End for
End if
End for

```

Siempre que se haya inicializado este array de dos dimensiones, se pueden obtener los punteros a los campos de una tabla concreta de la siguiente manera:

```

// Obtener los punteros a los campos para la tabla que se muestra actualmente en la pantalla:
COPY ARRAY(<>apFields{Table(Current form table)};$apTheFieldsIamWorkingOn)
// Inicializar los campos booleanos y de fecha
For($vlElem;1;Size of array($apTheFieldsIamWorkingOn))
    Case of
        :(Type($apTheFieldsIamWorkingOn{$vlElem}->)=Is date)
            $apTheFieldsIamWorkingOn{$vlElem}->:=Current date
        :(Type($apTheFieldsIamWorkingOn{$vlElem}->)=Is Boolean)
            $apTheFieldsIamWorkingOn{$vlElem}->:=True
    End case
End for

```

Nota: como sugiere este ejemplo, las líneas de un array de dos dimensiones pueden tener el mismo tamaño o diferentes tamaños.

## Arrays y memoria

A diferencia de los datos que se almacenan en el disco mediante tablas y registros, un array se mantiene siempre en memoria en su totalidad.

Por ejemplo, si se introdujeran todos los códigos postales de EE. UU. en la tabla [Zip Codes], ésta contendría unos 100.000 registros. Además, esa tabla incluiría varios campos: el propio código postal y la ciudad, el condado y el estado correspondientes. Si selecciona sólo los códigos postales de California, el motor de la base 4D crea la correspondiente selección de registros dentro de la tabla [Zip Codes], y luego carga los registros sólo cuando se necesitan (es decir, cuando se visualizan o imprimen). En palabras de orden, se trabaja con una serie ordenada de valores (del mismo tipo para cada campo) que se carga parcialmente desde el disco a la memoria por el motor de la base 4D.

Hacer lo mismo con arrays sería prohibido por las siguientes razones:

- Para mantener los cuatro tipos de información (código postal, ciudad, condado, estado), habría que mantener cuatro grandes arrays en memoria.
- Como un array se mantiene siempre en memoria en su totalidad, habría que mantener toda la información de los códigos postales en memoria durante toda la sesión de trabajo, aunque los datos no estén siempre en uso.
- De nuevo, dado que un array se mantiene siempre en memoria en su totalidad, cada vez que se inicia la aplicación

y se sale de ella, los cuatro arrays tendrían que cargarse y luego guardarse en el disco, aunque los datos no se utilicen ni se modifiquen durante la sesión de trabajo.

Conclusión: los arrays están pensados para mantener cantidades razonables de datos durante un corto periodo de tiempo. Por otro lado, como los arrays se mantienen en memoria, son fáciles de manejar y rápidos de manipular.

Sin embargo, en algunas circunstancias, puede ser necesario trabajar con arrays que contengan cientos o miles de elementos. La siguiente tabla muestra las fórmulas utilizadas para calcular la cantidad de memoria utilizada para cada tipo de array:

Tipo de array	Fórmula para determinar el uso de la memoria en bytes
Blob	(1+número de elementos) * 12 + Suma del tamaño de cada blob
Booleano	(31+número de elementos) * N8
Fecha	(1+número de elementos) * 6
Integer	(1+número de elementos) * 2
Entero largo	(1+número de elementos) * 4
Objeto	(1+número de elementos) * 8 + Suma del tamaño de cada objeto
Imagen	(1+número de elementos) * 8 + Suma del tamaño de cada imagen
Puntero	(1+número de elementos) * 8 + Suma del tamaño de cada puntero
Real	(1+número de elementos) * 8
Texto	(1+número de elementos) * 20 + (suma de la longitud de cada texto) * 2
Hora	(1+número de elementos) * 4
Dos dimensiones	(1+número de elementos) * 16 + Suma del tamaño de cada array

Notas:

- El tamaño de un texto en memoria se calcula con esta fórmula ((Longitud + 1) \* 2)
- Se requieren algunos bytes adicionales para llevar la cuenta del elemento seleccionado, el número de elementos y el propio array.

# Parámetros

A menudo encontrará que necesita pasar datos a sus métodos y funciones. Esto se hace fácilmente con parámetros.

## Generalidades

Los parámetros (o argumentos) son piezas de datos que un método o una función de clase necesita para realizar su tarea. Los términos *parámetros* y *argumentos* se utilizan indistintamente en este manual. Los parámetros también se pasan a los comandos integrados de 4D. En este ejemplo, la cadena "Hello" es un argumento para el comando integrado `ALERT` :

```
ALERT("Hello")
```

Los parámetros se pasan de la misma manera a los métodos o las funciones de clase. Por ejemplo, si una función de clase llamada `getArea()` acepta dos parámetros, una llamada a la función de clase podría verse así:

```
$area:=$o.getArea(50;100)
```

O, si un método proyecto llamado `DO_SOMETHING` acepta tres parámetros, una llamada al método podría verse así:

```
DO_SOMETHING($WithThis;$AndThat;$ThisWay)
```

Los parámetros de entrada están separados por punto y coma ( ; ).

Los mismos principios se aplican cuando los métodos se ejecutan a través de comandos dedicados, por ejemplo:

```
EXECUTE METHOD IN SUBFORM("Cal2";"SetCalendarDate";*;!05/05/20!)
//pase la fecha !05/05/20! como parámetro de SetCalendarDate
//en el contexto de un subformulario
//pase la fecha !05/05/20! como parámetro de SetCalendarDate
//en el contexto de un subformulario
```

Los datos también pueden ser devueltos desde métodos y funciones de clase. Por ejemplo, la siguiente línea de instrucción utiliza el comando integrado, `Length`, para devolver la longitud de una cadena. La instrucción pone el valor devuelto por `Length` en una variable llamada `MyLength`. Esta es la instrucción:

```
MyLength:=Length("How did I get here?")
```

Toda subrutina puede devolver un valor. Sólo se puede declarar un único parámetro de salida por método o función de clase.

Los valores de entrada y salida son [evaluados](#) en el momento de la llamada y copiados en variables locales dentro de la función o método de la clase llamada. Se proponen dos sintaxis para declarar los parámetros de las variables en el código llamado:

- [named variables](#) (recomendado en la mayoría de los casos) o
- [variables numeradas secuencialmente](#).

Las sintaxis [nombradas](#) y [secuenciales](#) se pueden combinar sin restricción para declarar los parámetros. Por ejemplo:

```
```4d
Function add($x : Integer)
  var $0;$2 : Integer
  $0:=$x+$2
```

## Parámetro con nombre

En los métodos y funciones de clase llamados, los valores de los parámetros se asignan a variables locales. Puedes declarar parámetros utilizando un nombre de parámetro con un tipo de parámetro, separados por dos puntos.

- Para las funciones de clase, los parámetros se declaran junto con la palabra clave `Function`.
- Para los métodos (métodos proyecto, métodos objeto formulario, métodos base y triggers), los parámetros se declaran utilizando la palabra clave `#DECLARE` al principio del código del método.

Ejemplos:

```
Function getArea($width : Integer; $height : Integer) -> $area : Integer
```

```
//myProjectMethod
#DECLARE ($i : Integer) -> $myResult : Object
```

Se aplican las siguientes reglas:

- La línea de declaración debe ser la primera línea del código del método o de la función, de lo contrario se mostrará un error (sólo los comentarios o los saltos de línea pueden preceder la declaración).
- Los nombres de los parámetros deben comenzar con un carácter `$` y cumplir con [reglas de denominación de las propiedades](#).
- Múltiples parámetros (y tipos) están separados por punto y coma (`;`).
- Las sintaxis multilínea están soportadas (utilizando el carácter `"$"`).

Por ejemplo, cuando se llama a una función `getArea()` con dos parámetros:

```
$area:=$o.getArea(50;100)
```

En el código de la función clase, el valor de cada parámetro se copia en el parámetro declarado correspondiente:

```
// Class: Polygon
Function getArea($width : Integer; $height : Integer)-> $area : Integer
  $area:=$width*$height
```

Si no se define el tipo, el parámetro se definirá como `Variant`.

Todos los tipos de métodos de 4D soportan la palabra clave `#DECLARE`, incluidos los métodos base. Por ejemplo, en el método base `On Web Authentication`, puede declarar parámetros temporales:

```
// Método base On Web Authentication
#DECLARE ($url : Text; $header : Text; \
  $BrowserIP : Text; $ServerIP : Text; \
  $user : Text; $password : Text) \
-> $RequestAccepted : Boolean
$entitySelection:=ds.User.query("login=:1"; $user)
// Verificar la contraseña hash...
```

## Valor devuelto

El parámetro de retorno de una función se declara añadiendo una flecha (->) y la definición del parámetro después de la lista de parámetros de entrada. Por ejemplo:

```
Function add($x : Variant; $y : Integer) -> $result : Integer
```

También se puede declarar el parámetro de retorno sólo añadiendo `: tipo`, en cuyo caso se puede manejar mediante una [instrucción de retorno](#) o a través de `$0` en la [sintaxis secuencial](#)). Por ejemplo:

```
Function add($x : Variant; $y : Integer): Integer  
$0:=$x+$y
```

## Tipos de datos soportados

Con los parámetros con nombre, puede utilizar los mismos tipos de datos [soportados por la palabra clave var](#), incluidos los objetos de las clases. Por ejemplo:

```
Function saveToFile($entity : cs.ShapesEntity; $file : 4D.File)
```

## Parámetros secuenciales

Como alternativa a la sintaxis [parámetros nombrados](#), puede declarar los parámetros utilizando variables numeradas secuencialmente: `$1`, `$2`, `$3`, etc. La numeración de las variables locales representa el orden de los parámetros.

Aunque esta sintaxis es soportada por las funciones clase, se recomienda utilizar la sintaxis [parámetros nombrados](#) en este caso.

Por ejemplo, cuando se llama a un método proyecto `DO_SOMETHING` con tres parámetros:

```
DO_SOMETHING($WithThis;$AndThat;$ThisWay)
```

En el código del método, el valor de cada parámetro se copia automáticamente en las variables `$1`, `$2`, `$3`:

```
//Código del método DO_SOMETHING  
//Asumiendo que todos los parámetros son de tipo texto  
C_TEXT($1;$2;$3)  
ALERT("I received "+$1+" and "+$2+" and also "+$3)  
//$1 contiene el parámetro $WithThis  
//$2 contiene el parámetro $AndThat  
//$3 contiene el parámetro $ThisWay
```

## Valor devuelto

El valor a devolver se pone automáticamente en la variable local `$0`.

Por ejemplo, el siguiente método, llamado `Uppercase4`, devuelve una cadena con los cuatro primeros caracteres de la cadena que se han pasado en mayúsculas:

```
$0:=Uppercase(Substring($1;1;4))+Substring($1;5)
```

El siguiente es un ejemplo que utiliza el método `Uppercase4`:

```
$NewPhrase:=Uppercase4("This is good.")
```

En este ejemplo, la variable `$NewPhrase` recibe "THIS is good."

El valor devuelto, `$0`, es una variable local dentro de la subrutina. Puede utilizarse como tal dentro de la subrutina. Por ejemplo, puede escribir:

```
// Do_something  
$0:=Uppercase($1)  
ALERT($0)
```

En este ejemplo, `$0` se le asignó primero el valor de `$1`, y luego se usó como parámetro del comando `ALERT`. Dentro de la subrutina, puede utilizar `$0` de la misma manera que utilizaría cualquier otra variable local. Es 4D quien devuelve el valor de `$0` (tal y como está cuando la subrutina termina) al método llamado.

## Tipos de datos soportados

Puede utilizar toda [expresión](#) como parámetro secuencial, excepto:

- tablas
- arrays

Las expresiones de tablas o arrays sólo pueden pasarse [como referencia utilizando un puntero](#).

## return {expression}

► Histórico

The `return` statement ends function or method execution and can be used to return an expression to the caller.

For example, the following function returns the square of its argument, `$x`, where `$x` is a number.

```
Function square($x : Integer)  
    return $x * $x
```

Internally, `return x` executes `$0:=x` or (if declared) `myReturnValue:=x`, and returns to the caller. If `return` is used without an expression, the function or method returns a null value of the declared return type (if any), otherwise *undefined*.

The `return` statement can be used along with the standard syntax for [returned values](#) (the returned value must be of the declared type). However, note that it ends immediately the code execution. Por ejemplo:

```
Function getValue  
    $0:=10  
    return 20  
    // devuelve 20  
  
Function getValue -> $v : Integer  
    return 10  
    $v:=20 // nunca se ejecuta  
    // devuelve 10
```

## Indirección de parámetros (\${N})

Los métodos proyecto 4D aceptan un número variable de parámetros. Puede dirigirse a esos parámetros con un bucle

`For...End for`, el comando `Count parameters` y la sintaxis de indirección de parámetros. Dentro del método, una dirección de indirección tiene el formato  `${N}`, donde `N` es una expresión numérica.  `${N}` se denomina parámetro genérico.

## Utilización de los parámetros genéricos

Por ejemplo, considere un método que suma valores y devuelve la suma formateada según un formato que se pasa como parámetro. Cada vez que se llama a este método, el número de valores a sumar puede variar. Debemos pasar los valores como parámetros al método y el formato en forma de cadena de caracteres. El número de valores puede variar de una llamada a otra.

Aquí está el método, llamado `MySum`:

```
#DECLARE($format : Text) -> $result : Text
$sum:=0
For($i;2;Count parameters)
    $sum:=$sum+$i
End for
$result:=String($sum;$format)
```

Los parámetros del método deben pasarse en el orden correcto, primero el formato y luego un número variable de valores:

```
Result:=MySum("##0.00";125,2;33,5;24) //"182.70"
Result:=MySum("000";1;2;200) //"203"
```

Tenga en cuenta que aunque haya declarado 0, 1 o más parámetros en el método, siempre puede pasar el número de parámetros que desee. Los parámetros están disponibles dentro del método llamado a través de la sintaxis  `${N}` y el tipo de los parámetros extra es `Variant` por defecto (puede declararlos utilizando una [directiva del compilador](#)). Sólo hay que asegurarse de que los parámetros existen, gracias al comando `Count parameters`. Por ejemplo:

```
//método foo
#DECLARE($p1: Text;$p2 : Text; $p3 : Date)
For($i;1;Count parameters)
    ALERT("param "+String($i)+" = "+String(${$i}))
End for
```

Este método se puede llamar:

```
foo("hello";"world";!01/01/2021!;42;?12:00:00?) //se pasan parámetros adicionales //se pasan parámetros
```

La indirección de parámetros se gestiona mejor si se respeta la siguiente convención: si sólo algunos de los parámetros se dirigen por indirección, deben pasarse después de los demás.

## Declaración de parámetros genéricos

Al igual que con otras variables locales, no es obligatorio declarar los parámetros genéricos mediante una directiva del compilador. Sin embargo, se recomienda para evitar toda ambigüedad. Los parámetros genéricos no declarados obtienen automáticamente el tipo `Variant`.

Para declarar parámetros genéricos, se utiliza una directiva del compilador a la que se pasa  `${N}` como parámetro, donde `N` especifica el primer parámetro genérico.

```
C_TEXT(${4})
```

La declaración de parámetros genéricos sólo puede hacerse con [la sintaxis secuencial](#).

Este comando significa que a partir del cuarto parámetro (incluido), el método puede recibir un número variable de parámetros de tipo texto. \$1, \$2 y \$3 pueden ser de cualquier tipo de datos. Sin embargo, si se utiliza \$2 por indirección, el tipo de datos utilizado será el tipo genérico. Así, será del tipo de datos texto, aunque para usted fuera, por ejemplo, del tipo de datos Real.

El número en la declaración tiene que ser una constante y no una variable.

## Declaración de los parámetros para el modo compilado

Aunque no sea obligatorio en [modo interpretado](#), debe declarar cada parámetro en los métodos o funciones llamados para evitar problemas.

Cuando se utiliza la [sintaxis de variables nombradas](#), los parámetros se declaran automáticamente a través de la palabra clave `#DECLARE` o del prototipo `Function`. Por ejemplo:

```
Function add($x : Variant; $y : Integer) -> $result : Integer  
    // todos los parámetros se declaran con su tipo
```

Cuando se utiliza [la sintaxis de la variable secuencial](#), hay que asegurarse de que todos los parámetros se declaran correctamente. En el siguiente ejemplo, el método `Capitalize` proyecto acepta un parámetro texto y devuelve un resultado texto:

```
// Método proyecto Mayusculas  
// Mayusculas( Texto ) -> Texto  
// Mayusculas( Cadena fuente ) -> Cadena con la primera letra en mayúscula  
  
C_TEXT($0;$1)  
$0:=Uppercase(Substring($1;1;1))+Lowercase(Substring($1;2))
```

La utilización de comandos tales como `New process` con métodos proceso que aceptan parámetros también requiere que los parámetros se declaren explícitamente en el método llamado. Por ejemplo:

```
C_TEXT($string)  
C_LONGINT($idProc;$int)  
C_OBJECT($obj)  
  
$idProc:=New process("foo_method";0;"foo_process";$string;$int;$obj)
```

Este código puede ser ejecutado en modo compilado sólo si "foo\_method" declara sus parámetros:

```
//foo_method  
C_TEXT($1)  
C_LONGINT($2)  
C_OBJECT($3)  
...
```

En modo compilado, puede agrupar todos los parámetros de las variables locales de los métodos proyecto en un método específico con un nombre que empiece por "Compiler". Dentro de este método, se pueden predeclarar

los parámetros de cada método, por ejemplo:

```
// Compiler_method  
C_REAL(OneMethodAmongOthers;$1)
```

Ver la página [Modos interpretado y compilado](#) para más información.

La declaración de parámetros también es obligatoria en los siguientes contextos (estos contextos no soportan la declaración en un método "Compiler"):

- Métodos base - Por ejemplo, el `método base On Web Connection` recibe seis parámetros, de \$1 a \$6, de tipo Texto. Al principio del método base, debe escribir (incluso si no se utilizan todos los parámetros):

```
// On Web Connection  
C_TEXT($1;$2;$3;$4;$5;$6)
```

También puede utilizar [parámetros nombrados](#) con la palabra clave `#DECLARE`.

- Triggers - El parámetro \$0 (Entero largo), que es el resultado de un trigger, será digitado por el compilador si el parámetro no ha sido declarado explícitamente. Sin embargo, si quiere declararlo, debe hacerlo en el propio trigger.
- Objetos formulario que aceptan el evento formulario `On Drag Over` - El parámetro \$0 (Entero largo), que es el resultado del evento formulario `On Drag Over`, será digitado por el compilador si el parámetro no ha sido declarado explícitamente. Sin embargo, si quiere declararlo, debe hacerlo en el propio método proyecto. Nota: el compilador no inicializa el parámetro \$0. Por lo tanto, tan pronto como utilice el evento formulario `On Drag Over`, debe inicializar \$0. Por ejemplo:

```
C_LONGINT($0)  
If(Form event=On Drag Over)  
  $0:=0  
  ...  
  If($DataType=Is picture)  
    $0:=-1  
  End if  
  ...  
End if
```

## Tipo de parámetro equivocado

Llamar a un parámetro con un tipo incorrecto es un [error](#) que impide la correcta ejecución. Por ejemplo, si escribe los siguientes métodos:

```
// method1  
#DECLARE($value : Text)
```

```
// method2  
method1(42) //tipo incorrecto, texto esperado
```

Este caso es tratado por 4D en función del contexto:

- en [proyectos compilados](#), se genera un error en el paso de compilación siempre que sea posible. En caso contrario, se genera un error cuando se llama al método.
- en los proyectos interpretados:

- o si el parámetro se declaró utilizando la sintaxis nombrada (`#DECLARE` o `Function`), se genera un error cuando se llama al método.
- o si el parámetro fue declarado utilizando la sintaxis secuencial (`c_XXX`), no se genera ningún error, el método llamado recibe un valor vacío del tipo esperado.

## VARIABLES DE ENTRADA/SALIDA

Dentro de la subrutina, puede utilizar los parámetros `$1`, `$2...` de la misma manera que utilizaría cualquier otra variable local. Sin embargo, en el caso de que utilice comandos que modifiquen el valor de la variable pasada como parámetro (por ejemplo `Find in field`), los parámetros `$1`, `$2`, etc. no pueden utilizarse directamente. Primero debe copiarlos en las variables locales estándar (por ejemplo: `$myvar:=$1`).

## UTILIZACIÓN DE LAS PROPIEDADES DE OBJETO COMO PARÁMETROS CON NOMBRE

La utilización de objetos como parámetros permite manejar parámetros con nombre. Este estilo de programación es simple, flexible y fácil de leer.

Por ejemplo, utilizando el método `CreatePerson`:

```
//CreatePerson
var $person : Object
$person:=New object("Name";"Smith";"Age";40)
ChangeAge($person)
ALERT(String($person.Age))
```

En el método `ChangeAge` puede escribir:

```
//ChangeAge
var $1; $para : Object
$para:=$1
$para.Age:=$para.Age+10
ALERT($para.Name+" is "+String($para.Age)+" years old.")
```

Esto ofrece una poderosa manera de definir parámetros opcionales (ver también abajo). Para manejar los parámetros que faltan, puede:

- verificar si se suministran todos los parámetros esperados comparándolos con el valor `Null`, o
- predefinir los valores de los parámetros, o
- utilizarlos como valores vacíos.

En el método `ChangeAge` anterior, las propiedades `Age` y `Name` son obligatorias y producirían errores si faltaran. Para evitar este caso, puede escribir simplemente:

```
//ChangeAge
var $1; $para : Object
$para:=$1
$para.Age:=Num($para.Age)+10
ALERT(String($para.Name)+" is "+String($para.Age)+" years old.")
```

Entonces ambos parámetros son opcionales; si no se llenan, el resultado será " is 10 years old", pero no se generará ningún error.

Por último, con los parámetros con nombre, el mantenimiento o la reproducción de las aplicaciones es muy sencillo y seguro. Imagine que más adelante se da cuenta de que añadir 10 años no siempre es apropiado. Necesita otro parámetro para definir cuántos años hay que añadir. Escriba:

```

$person:=New object("Name";"Smith";"Age";40;"toAdd";10)
ChangeAge($person)

//ChangeAge
var $1;$para : Object
$para:=$1
If ($para.toAdd=NULL)
    $para.toAdd:=10
End if
$para.Age:=Num($para.Age)+$para.toAdd
ALERT(String($para.Name)+" is "+String($para.Age)+" years old.")

```

El poder aquí es que no tendrá que cambiar su código existente. Siempre funcionará como en la versión anterior, pero si es necesario, puede utilizar otro valor que no sea 10 años.

Con las variables con nombre, cualquier parámetro puede ser opcional. En el ejemplo anterior, todos los parámetros son opcionales y se puede dar cualquiera, en cualquier orden.

## Parámetrosopcionales

En el manual *Lenguaje de 4D*, los caracteres { } (llaves) indican parámetros opcionales. Por ejemplo, `ALERT(message{}; okButtonTitle{})` significa que el parámetro *okButtonTitle* puede omitirse al llamar al comando. Se puede llamar de las siguientes maneras:

```

ALERT("Are you sure?";"Yes I am") //2 parámetros
ALERT("Time is over") //1 parámetro

```

Los métodos y las funciones 4D también aceptan estos parámetros opcionales. Tenga en cuenta que aunque haya declarado 0, 1 o más parámetros en el método, siempre puede pasar el número de parámetros que desee. Si llama a un método o función con menos parámetros que los declarados, los parámetros que faltan se procesan como valores por defecto en el código llamado, [según su tipo](#). Por ejemplo:

```

// función "concat" de myClass
Function concat ($param1 : Text ; $param2 : Text)->$result : Text
$result:=$param1+" "+$param2

```

```

// Método llamante
$class:=cs.myClass.new()
$class.concatenate("Hello") // "Hello "
$class.concatenate() // Displays " "

```

También puede llamar a un método o función con más parámetros de los declarados. Estarán disponibles en el código llamado a través de la sintaxis [\\${N}](#).

Utilizando el comando `Count parameters` desde dentro del método llamado, puede detectar el número real de parámetros y realizar diferentes operaciones dependiendo de lo que haya recibido.

El siguiente ejemplo muestra un mensaje de texto y puede insertar el texto en un documento en el disco o en un área de 4D Write Pro:

```

// APPEND TEXT Project Method
// APPEND TEXT ( Text { ; Text { ; Object } } )
// APPEND TEXT ( Message { ; Path { ; 4DWPArea } } )

Method($message : Text; $path : Text; $wpArea : Object)

ALERT($message)
If(Count parameters>=3)
    WP SET TEXT($wpArea;$1;wk append)
Else
    If(Count parameters>=2)
        TEXT TO DOCUMENT($path;$message)
    End if
End if

```

Después de añadir este método proyecto a su aplicación, puede escribir:

```

APPEND TEXT(vtSomeText) //Sólo mostrará el mensaje
APPEND TEXT(vtSomeText;$path) //Muestra el mensaje y el anexo al documento en $path
APPEND TEXT(vtSomeText;"";$wpArea) //Muestra el mensaje y lo escribe en $wpArea

```

Cuando los parámetros opcionales son necesarios en sus métodos, también puede considerar el uso de [propiedades de objeto como parámetros con nombre](#) que ofrecen una forma flexible de manejar un número variable de parámetros.

## Valores o referencias

Cuando pasa un parámetro, 4D siempre evalúa la expresión del parámetro en el contexto del método que llama y define el valor resultante en las variables locales en la función de clase o la subrutina. Las variables/parámetros locales no son los campos, variables o expresiones reales pasados por el método que llama; sólo contienen los valores que se han pasado. Las variables/parámetros locales no son los campos, variables o expresiones reales pasados por el método que llama; sólo contienen los valores que se han pasado. Por ejemplo:

```

//Esta es una parte del código del método MY_METHOD
DO_SOMETHING([People]Name) //Let's say [People]Name value is "williams"
ALERT([People]Name)

//Este es el código del método DO_SOMETHING
$1:=Uppercase($1)
ALERT($1)

```

La caja de alerta mostrada por `DO_SOMETHING` dirá "WILLIAMS" y la caja de alerta mostrada por `MY_METHOD` dirá "williams". El método cambió localmente el valor del parámetro \$1, pero esto no afecta al valor del campo `[People]Name` pasado como parámetro por el método `MY_METHOD`.

Hay dos formas de hacer que el método `DO_SOMETHING` cambie el valor del campo:

1. En lugar de pasar el campo al método, se pasa un puntero al mismo, por lo que se escribiría:

```

//Esta es una parte del código del método MY_METHOD
DO_SOMETHING(>[People]Name) //Let's say [People]Name value is "williams"
ALERT([People]Last Name)

//Este es el código del método DO_SOMETHING
$1->:=Uppercase($1->)
ALERT($1->)

```

Aquí el parámetro no es el campo, sino un puntero al mismo. Por lo tanto, dentro del método `DO SOMETHING`, \$1 ya no es el valor del campo sino un puntero al campo. El objeto referenciado por \$1 (\$1-> en el código anterior) es el campo real. Por lo tanto, cambiar el objeto referenciado va más allá del alcance de la subrutina, y el campo real se ve afectado. En este ejemplo, las dos cajas de alerta dirán "WILLIAMS".

2. En lugar de que el método `DO_SOMETHING` "haga algo", puede reescribir el método para que devuelva un valor. Por lo tanto, escribiría:

```
//Esta es una parte del código del método MY_METHOD
[People]Name:=DO_SOMETHING([People]Name) //Let's say [People]Name value is "williams"
ALERT([People]Name)

//Este es el código del método DO_SOMETHING
$0:=Uppercase($1)
ALERT($0)
```

Esta segunda técnica de retornar un valor por una subrutina se llama "utilizar una función" Se describe en el párrafo [Funciones](#). Se describe en el párrafo [Valores devueltos](#).

## Casos particulares: objetos y colecciones

Debe prestar atención al hecho de que los tipos de datos Objeto y Colección sólo pueden manejarse a través de una referencia (es decir, un *puntero* interno).

Por consiguiente, cuando se utilizan estos tipos de datos como parámetros, `$1, $2...` no contienen *valores* sino *referencias*. La modificación del valor de los parámetros `$1, $2...` dentro de la subrutina se propagará a cualquier lugar donde se utilice el objeto o colección fuente. Este es el mismo principio que para [los punteros](#), excepto que los parámetros `$1, $2...` no necesitan ser desreferenciados en la subrutina.

Por ejemplo, considere el método `CreatePerson` que crea un objeto y lo envía como parámetro:

```
//CreatePerson
var $person : Object
$person:=New object("Name";"Smith";"Age";40)
ChangeAge($person)
ALERT(String($person.Age))
```

El método `ChangeAge` añade 10 al atributo Age del objeto recibido

```
//ChangeAge
#DECLARE ($person : Object)
$person.Age:=$person.Age+10
ALERT(String($person.Age))
```

Cuando se ejecuta el método `CreatePerson`, las dos cajas de alerta dirán "50" ya que la misma referencia de objeto es manejada por ambos métodos.

4D Server: cuando se pasan parámetros entre métodos que no se ejecutan en la misma máquina (utilizando por ejemplo la opción "Ejecutar en el servidor"), las referencias no son utilizables. En estos casos, se envían copias de los parámetros de objetos y colecciones en lugar de referencias.

# Objetos y colecciones compartidos

Los objetos compartidos y las colecciones compartidas son [objetos](#) y [colecciones](#) específicas cuyo contenido se comparte entre procesos. A diferencia de las [variables interproceso](#), los objetos compartidos y las colecciones compartidas tienen la ventaja de ser compatibles con los procesos 4D apropiativos: pueden pasarse por referencia como parámetros a comandos como `New process` o `CALL WORKER`.

Los objetos compartidos y las colecciones compartidas pueden almacenarse en variables declaradas con los comandos estándar `C_OBJECT` y `C_COLLECTION`, pero deben instanciarse utilizando comandos específicos:

- para crear un objeto compartido, utilice el comando `New shared object`,
- para crear una colección compartida, utilice el comando `New shared collection`.

Nota: los objetos y colecciones compartidos pueden definirse como propiedades de objetos o colecciones estándar (no compartidos).

Para modificar un objeto/colección compartido, se debe llamar a la estructura `Use...End use`. La lectura de un valor de objeto/colección compartido no requiere `Use...End use`.

Un catálogo único y global devuelto por el comando `Storage` está siempre disponible en toda la aplicación y sus componentes, y puede utilizarse para almacenar todos los objetos y colecciones compartidos.

## Utilización de objetos o colecciones compartidos

Una vez instanciado con los comandos `Nuevo objeto compartido` o `Nueva colección compartida`, las propiedades y elementos del objeto compartido/colección pueden ser modificados o leídos desde cualquier proceso de la aplicación.

### Modificación

Las siguientes modificaciones pueden efectuarse en objetos y colecciones compartidos:

- añadir o eliminar propiedades de los objetos,
- añadir o editar valores (siempre que se soporten en objetos compartidos), incluyendo otros objetos compartidos o colecciones (lo que crea un grupo compartido, ver abajo).

Sin embargo, todas las instrucciones de modificación en un objeto compartido o colección deben estar rodeadas por las palabras clave `Use...End use`, de lo contrario se genera un error.

```
$s_obj:=New shared object("prop1";"alpha")
Use($s_obj)
  $s_obj.prop1:="omega"
End Use
```

Un objeto/una colección compartido(a) sólo puede modificarse por un proceso a la vez. `Use` bloquea el objeto/colección compartida para los otros hilos, mientras que `End use` desbloquea el objeto/colección compartida (si el contador de bloqueo está a 0, ver más adelante). Intentar modificar un objeto/colección compartido sin al menos un `Use...End use` genera un error. Cuando un proceso llama a `Use...End use` en un objeto/colección compartido que ya está en uso por otro proceso, simplemente se pone en espera hasta que el `End use` lo desbloquee (no se genera ningún error). En consecuencia, las instrucciones dentro de las estructuras `Use...End use` deben ejecutarse rápidamente y desbloquear los elementos lo antes posible. Por lo tanto, se recomienda enfáticamente evitar modificar un objeto o colección compartido directamente desde la interfaz, por ejemplo, a través de una caja de diálogo.

La asignación de objetos/colecciones compartidos a propiedades o elementos de otros objetos/colecciones compartidos está permitida y crea grupos compartidos. Un grupo compartido se crea automáticamente cuando un objeto/colección compartido se define como valor de propiedad o elemento de otro objeto/colección compartido. Los grupos compartidos permiten anidar objetos y colecciones compartidos, pero imponen reglas adicionales:

- Al llamar a `Use` en un objeto/colección compartido que pertenece a un grupo se bloquean las

propiedades/elementos de todos los objetos/colecciones del grupo y se incrementa su conteo de bloqueo. La llamada a `End use` disminuye el contador de bloqueo del grupo y cuando el contador está a 0, todos los objetos/colecciones compartidos vinculados se desbloquean.

- Un objeto/colección compartido sólo puede pertenecer a un grupo compartido. Se devuelve un error si se intenta asignar un objeto/colección compartido ya agrupado a un grupo diferente.
- Los objetos/colecciones compartidos agrupados no se pueden desagrupar. Una vez incluido en un grupo compartido, un objeto/colección compartido queda vinculado permanentemente a ese grupo durante toda la sesión. Incluso si todas las referencias de un objeto/colección se eliminan del objeto/colección padre, seguirán vinculadas.

Consulte el ejemplo 2 para ver una ilustración de las reglas de los grupos compartidos.

Nota: Los grupos compartidos se gestionan a través de una propiedad interna llamada *locking identifier*. Para obtener información detallada sobre este valor, consulte la guía del desarrollador de 4D.

## Lectura

Se permite la lectura de propiedades o elementos de un objeto/colección compartida sin tener que llamar a la estructura `Use...End use`, incluso si el objeto/colección compartida está en uso por otro proceso.

Sin embargo, cuando varios valores son interdependientes y deben ser leídos simultáneamente, es necesario enmarcar el acceso de lectura con una estructura `Use...End use` por coherencia.

## Duplication

Llamar a `OB Copy` con un objeto compartido (o con un objeto cuyas propiedades son objetos compartidos) es posible, pero en este caso se devuelve un objeto estándar (no compartido).

## Storage

Storage es un objeto compartido único, disponible automáticamente en cada aplicación y máquina. Este objeto compartido es devuelto por el comando `Storage`. Puede utilizar este objeto para hacer referencia a todos los objetos/colecciones compartidos definidos durante la sesión que deseé que estén disponibles desde cualquier proceso preventivo o estándar.

Tenga en cuenta que, a diferencia de los objetos compartidos estándar, el objeto `Storage` no crea un grupo compartido cuando se añaden objetos/colecciones compartidos como sus propiedades. Esta excepción permite utilizar el objeto `Storage` sin bloquear todos los objetos o colecciones compartidos conectados.

Para más información, consulte la descripción del comando `Storage`.

## Use...End use

La sintaxis de la estructura `Use...End use` es:

```
Use(Shared_object_or_Shared_collection)
    instrucción(es)
End use
```

La estructura `Use...End use` define una secuencia de instrucciones que ejecutarán tareas sobre el parámetro `Shared_object_or_Shared_collection` bajo la protección de un semáforo interno. `Shared_object_or_Shared_collection` puede ser cualquier objeto o colección compartido válido.

Los objetos compartidos y las colecciones compartidas están diseñados para permitir la comunicación entre procesos, en particular, procesos 4D preferentes. Se pueden pasar por referencia como parámetros de un proceso a otro. Para obtener información detallada sobre los objetos compartidos o las colecciones compartidas, consulte la página Objetos y colecciones compartidos. Es obligatorio rodear las modificaciones en los objetos o colecciones compartidas con las palabras clave `Use...End use` para evitar el acceso concurrente entre procesos.

- Una vez que se ejecuta con éxito la línea `Use`, todas las propiedades/elementos de `Shared_object_or_Shared_collection` se bloquean para el resto de procesos en acceso de escritura hasta que se

ejecute la línea `End use` correspondiente.

- La secuencia de *instrucciones* puede ejecutar cualquier modificación en las propiedades/elementos de `Shared_object_o_Shared_collection` sin riesgo de acceso concurrente.
- Si se añade otro objeto o colección compartida como propiedad del parámetro `Shared_object_or_Shared_collection`, se conectan dentro del mismo grupo compartido (ver Uso de objetos o colecciones compartidos).
- Si otro proceso intenta acceder a una de las propiedades `Objeto_compartido_o_Colección_compartida` o una propiedad conectada mientras se está ejecutando una secuencia `Use...` `End use`, se pone automáticamente en espera y espera hasta que la secuencia actual finalice.
- La línea `End use` desbloquea las propiedades `Shared_object_or_Shared_collection` y todos los objetos del mismo grupo.
- En el código 4D se pueden anidar varias estructuras `Use...End use`. En el caso de un grupo, cada `Use` incrementa el contador de bloqueo del grupo y cada `End use` lo disminuye; todas las propiedades/elementos se liberarán sólo cuando la última llamada `End use` ponga el contador de bloqueo a 0.

Nota: si un método de una colección modifica una colección compartida, se llama automáticamente un `Use interno` para esta colección compartida mientras se ejecuta la función.

## Ejemplo 1

Se desea lanzar varios procesos que realicen una tarea de inventario en diferentes productos y que actualicen el mismo objeto compartido. El proceso principal instancia un objeto compartido vacío y luego, lanza los otros procesos, pasando el objeto compartido y los productos a contar como parámetros:

```
ARRAY TEXT($_items;0)
... //llenar el array con los elementos a contar
$nbItems:=Size of array($_items)
C_OBJECT($inventory)
$inventory:=New shared object
Use($inventory)
    $inventory.nbItems:=$nbItems
End use

//Crear procesos
For($i;1;$nbItems)
    $ps:=New process("HowMany";0;"HowMany_"+$_items{$i};$_items{$i};$inventory)
    //Inventory object sent by reference
End for
```

En el método "HowMany", el inventario se realiza y el objeto compartido `$inventory` se actualiza lo antes posible:

```
C_TEXT($1)
C_TEXT($what)
C_OBJECT($2)
C_OBJECT($inventory)
$what:=$1 //para una mejor legibilidad
$inventory:=$2

$count:=CountMethod($what) //método para contar productos
Use($inventory) //utilizar el objeto compartido
    $inventory[$what]:=$count //guardar los resultados de este artículo
End use
```

## Ejemplo 2

Los siguientes ejemplos ilustran las reglas específicas para el manejo de los grupos compartidos:

```
$ob1:=New shared object
$ob2:=New shared object
Use($ob1)
  $ob1.a:=$ob2 //se crea el grupo 1
End use

$ob3:=New shared object
$ob4:=New shared object
Use($ob3)
  $ob3.a:=$ob4 //se crea el grupo 2
End use

Use($ob1) //utilizar un objeto del grupo 1
  $ob1.b:=$ob4 //ERROR
//$ob4 ya pertenece a otro grupo
//la asignación no está permitida
End use

Use($ob3)
  $ob3.a:=Null //eliminar cualquier referencia a $ob4 del grupo 2
End use

Use($ob1) //utilizar un objeto del grupo 1
  $ob1.b:=$ob4 //ERROR
//$ob4 aún pertenece al grupo 2
//la asignación no está permitida
End use
```

# Clases

## Generalidades

El lenguaje 4D soporta el concepto de clases. En un lenguaje de programación, el uso de una clase permite definir el comportamiento de un objeto con propiedades y funciones asociadas.

Cada objeto es una instancia de su clase. Una vez definida una clase usuario, puede instanciar los objetos de esta clase en cualquier parte de su código. Una clase puede extenderse a otra clase con la palabra clave `extender` y entonces hereda sus `funciones` y sus propiedades (`static` y `computed`).

El modelo de clases en 4D es similar al de las clases en JavaScript, y se basa en una cadena de prototipos.

Por ejemplo, puede crear una clase `Person` con la siguiente definición:

```
//Class: Person.4dm
Class constructor($firstname : Text; $lastname : Text)
    This.firstName:=$firstname
    This.lastName:=$lastname

Function get fullName() -> $fullName : text
    $fullName:=This.firstName+" "+This.lastName

Function sayHello()->$welcome : Text
    $welcome:="Hello "+This.fullName
```

En un método, creando una "Persona":

```
var $person : cs.Person //object of Person class
var $hello : Text
$person:=cs.Person.new("John";"Doe")
// $person:{firstName: "John"; lastName: "Doe"; fullName: "John Doe"}
$hello:=$person.sayHello() //"Hello John Doe"
```

## Gestión de clases

### Definición de una clase

Una clase usuario en 4D está definida por un archivo de método específico (.4dm), almacenado en la carpeta `/Project/Sources/Classes/`. El nombre del archivo es el nombre de la clase.

Al nombrar las clases, debe tener en cuenta las siguientes reglas:

- Un `nombre de clase` debe cumplir con `reglas de denominación de las propiedades`.
- Los nombres de clases son sensibles a las mayúsculas y minúsculas.
- No se recomienda dar el mismo nombre a una clase y a una tabla de la base, para evitar conflictos.

Por ejemplo, si quiere definir una clase llamada "Polygon", tiene que crear el siguiente archivo:

- Carpeta Project
  - Project

```
* Sources
  - Clases
    + Polygon.4dm
```

## Borrar una clase

Para eliminar una clase existente, puede:

- en su disco, elimine el archivo de clase .4dm de la carpeta "Clases",
- en el Explorador 4D, seleccionar la clase y hacer clic en o elegir Mover a la papelera en el menú contextual.

## Utilizar la interfaz 4D

Los archivos de clase se almacenan automáticamente en la ubicación adecuada cuando se crean a través de la interfaz de 4D, ya sea a través del menú Archivo o del Explorador.

### Menú Archivo y barra de herramientas

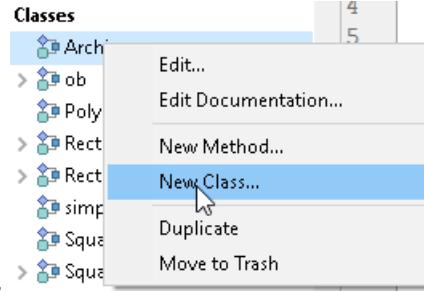
Puede crear un nuevo archivo de clase para el proyecto seleccionando Nuevo > Clase... en el menú Archivo de 4D Developer o en la barra de herramientas.

También puede utilizar el atajo Ctrl+Mayús+Alt+k.

### Explorador

En la página Métodos del Explorador, las clases se agrupan en la categoría Clases.

Para crear una nueva clase, puede:

- seleccione la categoría Clases y haga clic en el botón .
- seleccione Nueva clase... en el menú de acciones de la parte inferior de la ventana del Explorador, o en el menú contextual del grupo Clases.  

- seleccione Nueva > Clase... en el menú contextual de la página de inicio del Explorador.

### Soporte del código de clase

En las diferentes ventanas 4D (editor de código, compilador, depurador, explorador de ejecución), el código de la clase se maneja básicamente como un método proyecto con algunas especificidades:

- En el editor de código:
  - una clase no puede ser ejecutada
  - una función de clase es un bloque de código
  - Ir a la definición en un objeto miembro busca las declaraciones de función de clase; por ejemplo, "\$o.f()" encontrará "Function f".
  - Buscar referencias en la declaración de función de clase busca la función utilizada como miembro de objeto; por ejemplo, "Function f" encontrará "\$o.f()".
- En el explorador de ejecución y el depurador, las funciones de clase se muestran con el formato <ClassName> constructor o <ClassName>. <FunctionName> formato.

## Class stores

Las clases disponibles son accesibles desde sus class stores. Hay dos class stores disponibles:

- `cs` para el class store usuario
- `4D` para el class store integrado

## CS

`cs` -> `classStore`

Parámetros	Tipo		Descripción
<code>classStore</code>	objeto	<-	Class store usuario para el proyecto o componente

El comando `cs` devuelve la class store usuario para el proyecto o componente actual. Devuelve todas las clases de usuario [definidas](#) en el proyecto o componente abierto. Por defecto, sólo las [clases ORDA](#) están disponibles.

## Ejemplo

Quiere crear una nueva instancia de un objeto de `myClass` :

```
$instance:=cs.myClass.new()
```

## 4D

`4D` -> `classStore`

Parámetros	Tipo		Descripción
<code>classStore</code>	objeto	<-	Class store 4D

El comando `4D` devuelve la class store 4D integrada disponible. Ofrece acceso a APIs específicas como [CryptoKey](#).

## Ejemplo

Quiere crear una nueva llave en la clase `CryptoKey` :

```
$key:=4D.CryptoKey.new(New object("type";"ECDSA";"curve";"prime256v1"))
```

## El objeto clase

Cuando una clase es [definida](#) en el proyecto, se carga en el entorno del lenguaje 4D. Una clase es un objeto de la [clase "Class"](#). Una clase es un objeto en sí mismo, de ["Class" class](#).

- String `name`
- objeto `superclass` (null si ninguno)
- función `new()`, que permite instanciar objetos de clase.

Además, un objeto clase puede hacer referencia a un objeto `constructor` (opcional).

Un objeto de clase es un [objeto compartido](#) y, por tanto, se puede acceder a él desde diferentes procesos de 4D simultáneamente.

## Herencia

Si una clase hereda de otra clase (es decir, se utiliza la palabra clave [Class extends](#) en su definición), la clase padre es su `superclass`.

Cuando 4D no encuentra una función o una propiedad en una clase, la busca en su `superclass`; si no la encuentra, 4D sigue buscando en la superclase de la superclase, y así sucesivamente hasta que no haya más superclase (todos los

objetos heredan de la superclase "Object").

## Palabras clave de clase

En las definiciones de clase se pueden utilizar palabras claves específicas de 4D:

- `Function <Name>` para definir las funciones de clase de los objetos.
- `Function get <Name>` y `Function set <Name>` para definir las propiedades calculadas de los objetos.
- `Class constructor` para definir las propiedades estáticas de los objetos.
- `Class extends <ClassName>` para definir la herencia.

### Function

Sintaxis

```
Function <name>({$parameterName : type; ...}){->$parameterName : type}  
// code
```

Las funciones de clase son propiedades específicas de la clase. Son objetos de la clase [4D.Function](#).

En el archivo de definición de clase, las declaraciones de función utilizan la palabra clave `Function`, y el nombre de la función. El nombre de la función debe cumplir con las [reglas de nomenclatura de las propiedades](#).

Consejo: comenzar el nombre de la función con un carácter de subrayado ("\_") excluirá la función de las funcionalidades de autocompletado en el editor de código 4D. Por ejemplo, si declara `Function _myPrivateFunction` en `MyClass`, no se propondrá en el editor de código cuando digite en `"cs.MyClass. "`.

Inmediatamente después del nombre de la función, los [parámetros](#) de la función se pueden declarar con un nombre y un tipo de datos asignados, incluido el parámetro de retorno (opcional). Por ejemplo:

```
Function computeArea($width : Integer; $height : Integer)->$area : Integer
```

En una función de clase, el comando `This` se utiliza como instancia del objeto. Por ejemplo:

```
Function setFullscreen($firstname : Text; $lastname : Text)  
    This.firstname:=$firstname  
    This.lastname:=$lastname  
  
Function getFullscreen()->$fullname : Text  
    $fullname:=This.firstname+" "+Uppercase(This.lastname)
```

For a class function, the `Current method name` command returns: `<ClassName>.<FunctionName>`, for example `"MyClass.myMethod"`.

In the application code, class functions are called as member methods of the object instance and can receive [parameters](#) if any. Se soportan las siguientes sintaxis:

- utilización del operador `()`. For example, `myObject.methodName("hello")`
- utilización de un método miembro de la clase "4D.Function":
  - `apply()`
  - `call()`

Thread-safety warning: If a class function is not thread-safe and called by a method with the "Can be run in preemptive process" attribute: - the compiler does not generate any error (which is different compared to regular methods), - an error is thrown by 4D only at runtime.

## Parámetros

Los parámetros de las funciones se declaran utilizando el nombre del parámetro y su tipo, separados por dos puntos. El nombre del parámetro debe cumplir con las [reglas de nomenclatura de las propiedades](#). Múltiples parámetros (y tipos) están separados por punto y coma (;).

```
Function add($x; $y : Variant; $z : Integer; $xy : Object)
```

Si no se declaró el tipo, el parámetro se definirá como `Variant`.

The [classic 4D syntax](#) for method parameters can be used to declare class function parameters. Ambas sintaxis pueden mezclarse. Por ejemplo:

```
Function add($x : Integer)
  var $2; $value : Integer
  var $0 : Text
  $value:=$x+$2
  $0:=String($value)
```

## Valor devuelto

Se declara el parámetro de retorno (opcional) añadiendo una flecha (`->`) y la definición del parámetro de retorno después de la lista de parámetros de entrada, o dos puntos (`:`) y el tipo de parámetro de retorno únicamente. Por ejemplo:

```
Function add($x : Variant; $y : Integer)->$result : Integer
  $result:=$x+$y
```

También puede declarar el parámetro de retorno añadiendo sólo `: type` y utilizar la expresión `retorno` (también terminará la ejecución de la función). Por ejemplo:

```
Function add($x : Variant; $y : Integer): Integer
  // algún código
  return $x+$y
```

## Ejemplo 1

```
// Class: Rectangle
Class constructor($width : Integer; $height : Integer)
  This.name:="Rectangle"
  This.height:=$height
  This.width:=$width

// Definición de función
Function getArea()->$result : Integer
  $result:=(This.height)*(This.width)
```

```
// En un método proyecto

var $rect : cs.Rectangle
var $area : Real

$rect:=cs.Rectangle.new(50;100)
$area:=$rect.getArea() //5000
```

## Ejemplo 2

Este ejemplo utiliza la [expresión retorno](#) :

```
Function getRectArea($width : Integer; $height : Integer) : Integer
    If ($width > 0 && $height > 0)
        return $width * $height
    Else
        return 0
    End if
```

## Function get y Function set

### Sintaxis

```
Function get <name>()->$result : type
// código
```

```
Function set <name>($parameterName : type)
// código
```

`Function get` y `Function set` son accesos que definen las propiedades calculadas en la clase. Una propiedad calculada es una propiedad nombrada con un tipo de datos que enmascara un cálculo. Cuando se accede a un valor de propiedad calculado, 4D sustituye el código del accesor correspondiente:

- cuando se lee la propiedad, `Function get` se ejecuta,
- cuando se escribe la propiedad, `Function get` se ejecuta.

Si no se accede a la propiedad, el código nunca se ejecuta.

Las propiedades calculadas están diseñadas para manejar datos que no necesitan ser guardados en memoria. Generalmente se basan en propiedades persistentes. For example, if a class object contains as persistent property the *gross price* and the *VAT rate*, the *net price* could be handled by a computed property.

In the class definition file, computed property declarations use the `Function get` (the *getter*) and `Function set` (the *setter*) keywords, followed by the name of the property.

In the class definition file, computed property declarations use the `Function get` (the *getter*) and `Function set` (the *setter*) keywords, followed by the name of the property.

En el archivo de definición de la clase, las declaraciones de propiedades calculadas utilizan las palabras claves `Function get` (*getter*) y `Function set` (*setter*) seguido por el nombre de la propiedad. El nombre debe cumplir con las [reglas de nomenclatura de las propiedades](#).

`Función get` devuelve un valor del tipo de la propiedad y `Function set` toma un parámetro del tipo de la propiedad. Ambos argumentos deben cumplir con los [parámetros de función](#) estándar.

Cuando ambas funciones están definidas, la propiedad calculada es read-write. Si solo se define una `Function get`, la propiedad calculada es de solo lectura. En este caso, se devuelve un error si el código intenta modificar la propiedad. If

only a `Function set` is defined, 4D returns *undefined* when the property is read.

The type of the computed property is defined by the `$return` type declaration of the *getter*.

The type of the computed property is defined by the `$return` type declaration of the *getter*.

El tipo de la propiedad calculada es definido por la declaración de tipo `$return` del \*getter\*. Puede ser de cualquier tipo de propiedad válido.

Asignar *undefined* a una propiedad de objeto limpia su valor mientras se preserva su tipo. Para ello, la `Function get` es llamada primero para recuperar el tipo de valor, luego `Function set` es llamado con un valor vacío de ese tipo.

## Ejemplo 1

```
//Class: Person.4dm

Class constructor($firstname : Text; $lastname : Text)
    This.firstName:=$firstname
    This.lastName:=$lastname

Function get fullName() -> $fullName : Text
    $fullName:=This.firstName+" "+This.lastName

Function set fullName( $fullName : Text )
    $p:=Position(" "; $fullName)
    This.firstName:=Substring($fullName; 1; $p-1)
    This.lastName:=Substring($fullName; $p+1)
```

```
//en un método proyecto
$fullName:=$person.fullName // Function get fullName() is called
$person.fullName:="John Smith" // Function set fullName() is called
```

## Ejemplo 2

```
Function get fullAddress()->$result : Object
    $result:=New object

    $result.fullName:=This.fullName
    $result.address:=This.address
    $result.zipCode:=This.zipCode
    $result.city:=This.city
    $result.state:=This.state
    $result.country:=This.country
```

## Class Constructor

### Sintaxis

```
// Class: MyClass
Class Constructor({$parameterName : type; ...})
// code
// code
```

A class constructor function, which can accept [parameters](#), can be used to define a user class.

In that case, when you call the `new()` function, the class constructor is called with the parameters optionally passed to the `new()` function.

The type of the computed property is defined by the `$return` type declaration of the *getter*.

Ejemplo:

```
// Class: MyClass  
// Class constructor de MyClass  
Class Constructor ($name : Text)  
    This.name:=$name
```

```
// En un método proyecto  
// Se puede instanciar un objeto  
var $o : cs.MyClass  
$o:=cs.MyClass.new("HelloWorld")  
// $o = {"name":"HelloWorld"}
```

## Class extends <ClassName>

Sintaxis

```
// Class hijo  
Class extends <ParentClass>
```

The `Class extends` keyword is used in class declaration to create a user class which is a child of another user class. The child class inherits all functions of the parent class.

La extensión de clase debe respetar las siguientes reglas:

- A user class cannot extend a built-in class (except 4D.Object which is extended by default for user classes)
- A user class cannot extend a user class from another project or component.
- Una clase usuario no puede extenderse a sí misma.
- It is not possible to extend classes in a circular way (i.e. "a" extends "b" that extends "a").

Breaking such a rule is not detected by the code editor or the interpreter, only the compiler and `check syntax` will throw an error in this case.

An extended class can call the constructor of its parent class using the `Super` command.

Ejemplo

This example creates a class called `Square` from a class called `Polygon`.

```

//Class: Square

//path: Classes/Square.4dm

Class extends Polygon

Class constructor ($side : Integer)

    // It calls the parent class's constructor with lengths
    // provided for the Polygon's width and height
    Super($side;$side)
    // In derived classes, Super must be called before you
    // can use 'This'
    This.name:="Square"

Function getArea()
    C_LONGINT($0)
    $0:=This.height*This.width

```

## Super

### Sintaxis

```
Super {{ param{;...;paramN} }} {-> Object}
```

Parámetros	Tipo		Descripción
param	mixto	->	Parámetro(s) a pasar al constructor de la clase padre
Resultado	objeto	<-	Padre del objeto

The `Super` keyword allows calls to the `superclass`, i.e. the parent class.

`Super` tiene dos propósitos diferentes:

1. Inside a `constructor code`, `Super` is a command that allows to call the constructor of the superclass. When used in a constructor, the `Super` command appears alone and must be used before the `This` keyword is used.
- If all class constructors in the inheritance tree are not properly called, error -10748 is generated. Es responsabilidad del desarrollador 4D asegurarse de que las llamadas sean válidas.
- If the `This` command is called on an object whose superclasses have not been constructed, error -10743 is generated.
- If `Super` is called out of an object scope, or on an object whose superclass constructor has already been called, error -10746 is generated.

```

// inside myClass constructor
var $text1; $text2 : Text
Super($text1) //calls superclass constructor with a text param
This.param:=$text2 // use second param

```

2. Inside a `class member function`, `Super` designates the prototype of the superclass and allows to call a function of the superclass hierarchy.

```

Super.doSomething(42) //calls "doSomething" function
//declared in superclasses

```

### Ejemplo 1

This example illustrates the use of `Super` in a class constructor. The command is called to avoid duplicating the constructor parts that are common between `Rectangle` and `Square` classes.

```
// Class: Rectangle
Class constructor($width : Integer; $height : Integer)
    This.name:="Rectangle"
    This.height:=$height
    This.width:=$width

Function sayName()
    ALERT("Hi, I am a "+This.name+".") 

// Function definition
Function getArea()
    var $0 : Integer
    $0:=(This.height)*(This.width)
```

```
//Class: Square

Class extends Rectangle

Class constructor ($side : Integer)

    // It calls the parent class's constructor with lengths
    // provided for the Rectangle's width and height
    Super($side;$side)
    // In derived classes, Super must be called before you
    // can use 'This'
    This.name:="Square"

Function getArea()
    C_LONGINT($0)
    $0:=This.height*This.width
```

## Ejemplo 2

This example illustrates the use of `Super` in a class member method. You created the `Rectangle` class with a function:

```
//Class: Rectangle

Function nbSides()
    var $0 : Text
    $0:="I have 4 sides"
```

You also created the `Square` class with a function calling the superclass function:

```
//Class: Square

Class extends Rectangle

Function description()
    var $0 : Text
    $0:=Super.nbSides()+" which are all equal"
```

Entonces puede escribir en un método proyecto:

```

var $square : Object
var $message : Text
$square:=cs.Square.new()
$message:=$square.description() //I have 4 sides which are all equal

```

## This

### Sintaxis

`This` → Object

Parámetros	Tipo		Descripción
Resultado	objeto	<-	Objeto actual

The `This` keyword returns a reference to the currently processed object. En 4D, se puede utilizar en [contextos diferentes](#).

In most cases, the value of `This` is determined by how a function is called. No se puede definir por asignación durante la ejecución, y puede ser diferente cada vez que se llame a la función.

When a formula is called as a member method of an object, its `This` is set to the object the method is called on. Por ejemplo:

```

$o:=New object("prop";42;"f";Formula(This.prop))
$val:=$o.f() //42

```

When a [class constructor](#) function is used (with the `new()` function), its `This` is bound to the new object being constructed.

```

//Class: ob

Class Constructor

    // Create properties on This as
    // desired by assigning to them
    This.a:=42

```

```

// en un método 4D
$o:=cs.ob.new()
$val:=$o.a //42

```

When calling the superclass constructor in a constructor using the [Super](#) keyword, keep in mind that `This` must not be called before the superclass constructor, otherwise an error is generated. Ver [este ejemplo](#).

In any cases, `This` refers to the object the method was called on, as if the method were on the object.

```

//Class: ob

Function f()
$0:=This.a+This.b

```

Entonces puede escribir en un método proyecto:

```
$o:=cs.ob.new()  
$o.a:=5  
$o.b:=3  
$val:=$o.f() //8
```

In this example, the object assigned to the variable \$o doesn't have its own *f* property, it inherits it from its class. Since *f* is called as a method of \$o, its `This` refers to \$o.

## Comandos de clases

Several commands of the 4D language allows you to handle class features.

### OB Class

OB Class ( object ) -> Object | Null

`OB Class` returns the class of the object passed in parameter.

### OB Instance of

OB Instance of ( object ; class ) -> Boolean

`OB Instance of` returns `true` if `object` belongs to `class` or to one of its inherited classes, and `false` otherwise.

# Condiciones y bucles

Regardless of the simplicity or complexity of a method or function, you will always use one or more of three types of programming structures. Las estructuras de programación determinan el flujo de ejecución, si se ejecutan y el orden de las líneas de instrucciones dentro de un método. Hay tres tipos de estructuras:

- **Secuencial:** una estructura secuencial es una estructura simple y lineal. Una secuencia es una serie de sentencias que 4D ejecuta una tras otra, de la primera a la última. Una instrucción de una línea, utilizada frecuentemente para los métodos de los objetos, es el caso más simple de una estructura secuencial. Por ejemplo:  
`[People]lastName:=Uppercase( [People]lastName)`
- **Branching:** una estructura de bifurcación permite que los métodos prueben una condición y tomen caminos alternativos, dependiendo del resultado. La condición es una expresión booleana, una expresión que evalúa TRUE o FALSE. Una estructura condicional es la estructura `If...Else...End if`, que dirige el flujo del programa a lo largo de uno de los dos caminos. La otra estructura condicional es la estructura `Case of... Else...End case`, que dirige el flujo del programa a una de las muchas alternativas.
- **Bucle:** cuando se escriben métodos, es muy común encontrarse con que se necesita que una secuencia de sentencias se repita un número de veces. Para hacer frente a esta necesidad, el lenguaje 4D ofrece las siguientes estructuras de bucle:
  - `While...End while`
  - `Repeat...Until`
  - `For...End for`
  - `End for each`

Los bucles se controlan de dos maneras: o bien hacen un bucle hasta que se cumpla una condición, o bien hacen un bucle un número determinado de veces. Cada estructura de bucle puede utilizarse de cualquier manera, pero los bucles `While` y los bucles `Repeat` son más apropiados para repetir hasta que se cumpla una condición, y los bucles `For` son más apropiados para hacer un bucle un número determinado de veces. `End for each` permite la mezcla en ambos sentidos y está diseñado para realizar bucles dentro de objetos y colecciones.

Nota: 4D permite anidar estructuras de programación hasta una "profundidad" de 512 niveles.

## return {expression}

### ► Histórico

The `return` statement can be called from anywhere. When a `return` statement is used in a function or method, the execution of the function or method is stopped. The remaining code is not executed and the control is returned to the caller.

The `return` statement can be used to `return a value` to the caller.

## Ejemplo

```
var $message : Text
var $i : Integer

While (True) //infinite loop
    $i:=$i+1
    $message+=String($i)+"A\r" // until 5
    logConsole($message)
    If ($i=5)
        return //stops the loop
    End if
    $message+=String($i)+"B\r" // until 4
    logConsole($message)
End while
$message+=String($i)+"C\r" //never executed
logConsole($message)

// 1A
// 1B
// 2A
// 2B
// 3A
// 3B
// 4A
// 4B
// 5A
```

# Estructuras condicionales

Una estructura de ramificación permite que los métodos prueben una condición y tomen caminos alternativos, en función del resultado.

## If...Else...End if

La sintaxis de la estructura condicional `If...Else...End if` es:

```
If(Boolean_Expression)
    statement(s)
Else
    statement(s)
End if
```

Tenga en cuenta que la parte `Else` es opcional; puede escribir:

```
If(Boolean_Expression)
    statement(s)
End if
```

La estructura `If...Else...End if` permite a su método elegir entre dos acciones, dependiendo de si una prueba (una expresión booleana) es TRUE o FALSE. Cuando la expresión booleana es TRUE, se ejecutan las sentencias que siguen inmediatamente a la prueba. Si la expresión booleana es FALSE, se ejecutan las instrucciones que siguen a la línea `Else`. El `Else` es opcional; si se omite `Else`, la ejecución continúa con la primera instrucción (si la hay) que sigue al `End if`.

Tenga en cuenta que la expresión booleana siempre se evalúa completamente. Considere en particular la siguiente prueba:

```
If(MethodA & MethodB)
    ...
End if
```

La expresión es TRUE sólo si los dos métodos son TRUE. Sin embargo, incluso si `MethodA` devuelve FALSE, 4D seguirá evaluando `MethodB`, lo que supone una pérdida de tiempo innútil. En este caso, es más interesante utilizar una estructura como:

```
If(MethodA)
    If(MethodB)
        ...
    End if
End if
```

El resultado es similar y `MethodB` se evalúa sólo si es necesario.

Note: The [ternary operator](#) allows writing one-line conditional expressions and can replace a full sequence of `If...Else` statements.

## Ejemplo

```

// Pedir al usuario que introduzca un nombre
$Find:=Request(Type a name)
Si(OK=1)
  QUERY([People];[People]LastName=$Find)
Else
  ALERT("No ha introducido un nombre.")
End if
End if
End if
End if

```

Consejo: la ramificación puede realizarse sin que las instrucciones se ejecuten en un caso u otro. Al desarrollar un algoritmo o una aplicación especializada, nada le impide escribir:

```

If(Boolean_Expression)
Else
  statement(s)
End if

```

O:

```

If(Boolean_Expression)
  statement(s)
Else
End if

```

## Case of... Else...End case

La sintaxis de la estructura condicional `Case of...Else...End case` es:

```

Case of
  :(Boolean_Expression)
    statement(s)
  :(Boolean_Expression)
    statement(s)

  .
  .
  .

  :(Boolean_Expression)
    statement(s)
Else
  statement(s)
End case

```

Tenga en cuenta que la parte `Else` es opcional; puede escribir:

```

Case of
  :(Boolean_Expression)
    statement(s)
  :(Boolean_Expression)
    statement(s)
  .
  .
  .
  :(Boolean_Expression)
    statement(s)
End case

```

Al igual que la estructura `If...Else...End if`, la estructura `Case of...Else...End case` también permite a su método elegir entre acciones alternativas. A diferencia de la estructura `If...Else...End`, la estructura `Case of...Else...End case` puede probar un número razonablemente ilimitado de expresiones booleanas y realizar una acción dependiendo de cuál sea TRUE.

Cada expresión booleana va precedida de dos puntos ( `:` ). Esta combinación de los dos puntos y la expresión booleana se llama un caso. Por ejemplo, la siguiente línea es un caso:

```
: (bValidate=1)
```

Sólo se ejecutarán las instrucciones que sigan al primer caso TRUE (y hasta el siguiente). Si ninguno de los casos es TRUE, no se ejecutará ninguna de las instrucciones (si no se incluye la parte `Else`).

Puede incluir una instrucción `Else` después del último caso. Si todos los casos son FALSE, se ejecutarán las instrucciones siguientes al `Else`.

## Ejemplo

Este ejemplo comprueba una variable numérica y muestra un cuadro de alerta con una palabra:

```

Case of
  :(vResult=1) //Probar si el número es 1
    ALERT("One.") //Si es 1, mostrar una alerta
  :(vResult=2) //Probar si el número es 2
    ALERT("Two.") Case of
      :(vResult=1) //Probar si el número es 1
        ALERT("One.") //Si es 1, mostrar una alerta
      :(vResult=2) //Probar si el número es 2
        ALERT("Two.") //Si es 2, mostrar una alerta
      :(vResult=3) //Probar si el número es 3
        ALERT("Three.") //Si es 3, mostrar una alerta
    Else //Si no es 1, 2 o 3, mostrar una alerta
      ALERT("It was not one, two, or three.")
    //statement(s)
End case //Si es 3, mostrar una alerta
Else //Si no es 1, 2 o 3, mostrar una alerta
  ALERT("It was not one, two, or three.")
//statement(s)
End case //Si es 1, mostrar una alerta
  :(vResult=2) //Probar si el número es 2
    ALERT("Two.") //Si es 2, mostrar una alerta
  :(vResult=3) //Probar si el número es 3
    ALERT("Three.") //Si es 3, mostrar una alerta
Else //Si no es 1, 2 o 3, mostrar una alerta
  ALERT("It was not one, two, or three.")
End case

```

Para comparar, aquí está la versión `If...Else...End if` del mismo método:

```
If(vResult=1) //Probar si el número es 1
    ALERT("One.") //Si es 1, mostrar una alerta
Else
    If(vResult=2) //Probar si el número es 2
        ALERT("Two.") //Si es 2, mostrar una alerta
    Else
        If(vResult=3) //Probar si el número es 3
            ALERT("Three.") //Si es 3, mostrar una alerta
    Else //Si no es 1, 2 o 3, mostrar una alerta
        ALERT("It was not one, two, or three.")
    End if
End if
End if
```

Recuerde que con una estructura `Case of... Else...End case`, sólo se ejecuta el primer caso TRUE. Aunque dos o más casos sean TRUE, sólo se ejecutarán las instrucciones que siguen al primer caso TRUE.

En consecuencia, cuando quiera implementar pruebas jerárquicas, debe asegurarse de que las declaraciones de condición que están más abajo en el esquema jerárquico aparezcan primero en la secuencia de pruebas. Por ejemplo, si se quiere procesar el caso simple (`vResult=1`) y el caso complejo (`vResult=1 & (vCondition#2)`) y se estructura el método de la siguiente manera: Por ejemplo, el siguiente código nunca verá detectada su última condición:

```
Case of
    :(vResult=1)
        ... //statement(s)
    :((vResult=1) & (vCondition#2)) //este caso nunca será detectado
        ... //statement(s)
End case
```

En el código anterior, la presencia de la segunda condición no se detecta, ya que la prueba "`vResult=1`" ramifica el código antes de cualquier otra prueba. Para que el código funcione correctamente, puedes escribirlo así:

```
Case of
    :((vResult=1) & (vCondition#2)) //este caso será detectado primero
        ... //statement(s)
    :(vResult=1)
        ... //statement(s)
End case
```

Además, si quiere implementar pruebas jerárquicas, puede considerar el uso de código jerárquico.

Consejo: la ramificación puede realizarse sin que las instrucciones se ejecuten en un caso u otro. Al desarrollar un algoritmo o una aplicación especializada, nada le impide escribir:

```
Case of
    :(Boolean_Expression)
    :(Boolean_Expression)
    ...
    :(Boolean_Expression)
        statement(s)
    Else
        statement(s)
End case
```

O:

```
Case of
:(Boolean_Expression)
:(Boolean_Expression)
    statement(s)
    ...
:(Boolean_Expression)
    statement(s)
Else
End case
```

O:

```
Case of
Else
    statement(s)
End case
```

# Estructuras repetitivas (bucles)

Las estructuras en bucle repiten una secuencia de instrucciones hasta que se cumple una condición o se alcanza un número de veces.

## While...End while

La sintaxis de la estructura condicional `While...End while` es:

```
While(Boolean_Expression)
    statement(s)
    {break}
    {continue}
End while
```

Un bucle `While...End while` ejecuta las instrucciones dentro del bucle mientras la expresión booleana sea TRUE. Comprueba la expresión booleana al inicio del bucle y no entra en el bucle si la expresión es FALSE.

Las instrucciones `break` y `continue` se [describen a continuación](#).

Es común inicializar el valor probado en la expresión booleana inmediatamente antes de entrar en el bucle `While...End while`. Inicializar el valor significa asignarle un contenido adecuado, normalmente para que la expresión booleana sea TRUE y `While...End while` ejecute el bucle.

El valor de la expresión booleana debe poder ser modificado por un elemento dentro del bucle, de lo contrario se ejecutará indefinidamente. El siguiente bucle continúa para siempre porque `NeverStop` es siempre TRUE:

```
NeverStop:=True
While(NeverStop)
End while
```

Si se encuentra en una situación de este tipo, en la que un método se ejecuta de forma incontrolada, puede utilizar las funciones de rastreo para detener el bucle y localizar el problema. Para más información sobre el seguimiento de un método, consulte la página [Gestión de errores](#).

## Ejemplo

```
CONFIRM("¿Añadir un nuevo registro?") //¿El usuario quiere añadir un registro?
While(OK=1) //Bucle mientras el usuario quiera
    ADD RECORD([aTable]) /Añadir un nuevo registro
End while //El bucle siempre termina con End while
```

En este ejemplo, el valor de la variable sistema `OK` es definido por el comando `CONFIRM` antes de que se inicie el bucle. Si el usuario hace clic en el botón OK de la caja de diálogo de confirmación, la variable del sistema `OK` toma el valor 1 y se inicia el bucle. En caso contrario, la variable del sistema `OK` toma el valor 0 y se omite el bucle. Una vez se inicia el bucle, el comando `ADD RECORD` permite continuar la ejecución del bucle porque se define la variable sistema `OK` en 1 cuando el usuario guarda el registro. Cuando el usuario cancela (no guarda) el último registro, la variable del sistema `OK` toma el valor 0 y el bucle se detiene.

## Repeat...Until

La sintaxis de la estructura condicional `Repeat...Until` es:

```

Repeat
    statement(s)
    {break}
    {continue}
Until(Boolean_Expression)

```

Un bucle `Repeat...Until` es similar a un bucle `While...End while`, excepto que comprueba la expresión booleana después del bucle en lugar de antes. Así, un bucle `Repeat...Until` siempre ejecuta el bucle una vez, mientras que si la expresión booleana es inicialmente False, un bucle `While...End while` no ejecuta el bucle en absoluto.

La otra diferencia con un bucle `Repeat...Until` es que el bucle continúa hasta que la expresión booleana sea TRUE.

Las instrucciones `break` y `continue` se [describen a continuación](#).

## Ejemplo

Compara el siguiente ejemplo con el ejemplo del bucle `While...End while`. Tenga en cuenta que la expresión booleana no necesita ser inicializada-no hay un comando `CONFIRM` para inicializar la variable `OK`.

```

Repeat
    ADD RECORD([aTable])
Until(OK=0)

```

## For...End for

La sintaxis de la estructura condicional `For...End for` es:

```

For(Counter_Variable;Start_Expression;End_Expression{;Increment_Expression})
    statement(s)
    {break}
    {continue}
End for

```

El bucle `For...End for` es un bucle controlado por un contador:

- La variable contador `Counter_Variable` es una variable numérica (Real o Entero largo) inicializada por `For...End for` con el valor especificado por `Start_Expression`.
- Cada vez que se ejecuta el bucle, la variable del contador se incrementa en el valor especificado en el valor opcional `Increment_Expression`. Si no se especifica `Increment_Expression`, la variable del contador se incrementa en uno (1), que es el valor predeterminado.
- Cuando la variable del contador pasa el valor `End_Expression`, el bucle se detiene.

Importante: las expresiones numéricas `Start_Expression`, `End_Expression` y `Increment_Expression` se evalúan una vez al principio del bucle. Si estas expresiones son variables, el cambio de una de estas variables dentro del bucle no afectará al bucle.

Consejo: Sin embargo, para fines especiales, puede cambiar el valor de la variable `Counter_Variable` dentro del bucle; esto afectará al bucle.

- Normalmente `Start_Expression` es menor que `End_Expression`.
- Si `Start_Expression` y `End_Expression` son iguales, el bucle se ejecutará sólo una vez.
- Si `Start_Expression` es mayor que `End_Expression`, el bucle no se ejecutará en absoluto a menos que se especifique una `Increment_Expression` negativa. Ver los ejemplos.

Las instrucciones `break` y `continue` se [describen a continuación](#).

## Ejemplos básicos

1. El siguiente ejemplo ejecuta 100 iteraciones:

```
For(vCounter;1;100)
  //Hacer algo
End for
```

2. El siguiente ejemplo recorre todos los elementos del array anArray:

```
For($vlElem;1;Size of array(anArray))
  //Hacer algo con el elemento
  anArray{$vlElem}:=...
End for
```

3. El siguiente ejemplo recorre todos los caracteres del texto vtSomeText:

```
For($vlChar;1;Length(vtSomeText))
  //Hacer algo con el carácter si es un TAB
  If(Character code(vtSomeText[$vlChar])=Tab)
    //...
  End if
End for
```

4. El siguiente ejemplo recorre los registros seleccionados para la tabla [aTable]:

```
FIRST RECORD([aTable])
For($vlRecord;1;Records in selection([aTable]))
  //Hacer algo con el registro
  SEND RECORD([aTable])
  //...
  //Ir al siguiente registro
  NEXT RECORD([aTable])
End for
```

La mayoría de los bucles `For...End for` que escribirá en sus proyectos se parecerán a los que se presentan en estos ejemplos.

## Disminuir la variable contador

En algunos casos, puede querer tener un bucle cuya variable de contador sea decreciente en lugar de creciente. Para ello, debe especificar *Start\_Expression* mayor que *End\_Expression* y *Increment\_Expression* debe ser negativa. Los siguientes ejemplos hacen lo mismo que los anteriores, pero en orden inverso:

5. El siguiente ejemplo ejecuta 100 iteraciones:

```
For(vCounter;100;1;-1)
  //Hacer algo
End for
```

6. El siguiente ejemplo recorre todos los elementos del array anArray:

```
For($vlElem;Size of array(anArray);1;-1)
  //Hacer algo con el elemento
  anArray{$vlElem}:=...
End for
```

7. El siguiente ejemplo recorre todos los caracteres del texto vtSomeText:

```
For($vlChar;Length(vtSomeText);1;-1)
  //Hacer algo con el carácter si es un TAB
  If(Character_code(vtSomeText[$vlChar])=Tab)
  //...
  End if
End for
```

8. El siguiente ejemplo recorre los registros seleccionados para la tabla [aTable]:

```
LAST RECORD([aTable])
For($vlRecord;1;Records in selection([aTable]))
  //Hacer algo con el registro
  SEND RECORD([aTable])
  //...
  //Ir al registro anterior
  PREVIOUS RECORD([aTable])
End for
```

## Incrementar la variable del contador en más de uno

Si lo requiere, puede utilizar una *Increment\_Expression* (positiva o negativa) cuyo valor absoluto sea mayor que uno.

9. El siguiente bucle aborda sólo los elementos pares del array anArray:

```
For($vlElem;2;Size of array(anArray);2)
  //Hacer algo con el elemento #2,#4...#2n
  anArray[$vlElem]:=...
End for
```

## Comparación de estructuras de bucle

Volvamos al primer ejemplo de `For...End for`. El siguiente ejemplo ejecuta 100 iteraciones:

```
For(vCounter;1;100)
  //Hacer algo
End for
```

Es interesante ver cómo el bucle `While...End while` y el bucle `Repeat...Until` realizarían la misma acción. Aquí está el bucle equivalente `While...End while`:

```
$i:=1 //Inicializar el contador
While($i<=100) //Bucle 100 veces
  //Hacer algo
  $i:=$i+1 //Necesita incrementar el contador
End while
```

Aquí está el bucle equivalente `Repeat...Until`:

```

$i:=1 //Inicializar el contador
Repeat
  //Hacer algo
  $i:=$i+1 //Necesita incrementar el contador
Until($i=100) //Bucle 100 veces

```

Consejo: el bucle `For...End for` suele ser más rápido que los bucles `While...End while` y `Repeat...Until`, porque 4D comprueba la condición internamente en cada ciclo del bucle e incrementa el contador. Por lo tanto, utilice el bucle `For...End for` siempre que sea posible.

## Optimizar la ejecución de los bucles For... End for

Puede utilizar variables reales y enteras, así como contadores interproceso, de proceso y de variables locales. Para bucles repetitivos largos, especialmente en modo compilado, utilice variables locales de tipo Entero largo.

10. Aquí un ejemplo:

```

C_LONGINT($vlCounter) //Utilización de variables locales de tipo Entero largo
For($vlCounter;1;10000)
  //Hacer algo
End for

```

## Estructuras For... End anidadas

Puede anidar tantas estructuras de control como necesite (razonablemente). Esto incluye la anidación de bucles `For...End for`. Para evitar errores, asegúrese de utilizar diferentes variables de contador para cada estructura de bucle.

He aquí dos ejemplos:

1. El siguiente ejemplo recorre todos los elementos de un array de dos dimensiones:

```

For($vlElem;1;Size of array(anArray))
  //...
  //Hacer algo con la línea
  //...
  For($vlSubElem;1;Size of array(anArray{$vlElem}))
    //Hacer algo con el elemento
    anArray{$vlElem}{$vlSubElem}:=...
  End for
End for

```

2. El siguiente ejemplo construye un array de punteros a todos los campos de fecha presentes en la base:

```

ARRAY POINTER($apDateFields;0)
$vlElem:=0
For($vlTable;1;Get last table number)
  If(Is table number valid($vlTable))
    For($vlField;1;Get last field number($vlTable))
      If(Is field number valid($vlTable;$vlField))
        $vpField:=Field($vlTable;$vlField)
        If(Type($vpField->)=Is date)
          $vlElem:=$vlElem+1
          INSERT IN ARRAY($apDateFields;$vlElem)
          $apDateFields{$vlElem}:=$vpField
        End if
      End if
    End for
  End if
End for

```

## For each... End for each

La sintaxis de la estructura condicional `For each... End for each` es:

```

For each(Current_Item;Expression{;begin{;end}}){Until|While}(Boolean_Expression)}
  statement(s)
  {break}
  {continue}
End for each

```

La estructura `For each... End for each` ejecuta un *Current\_item* especificado sobre todos los valores de *Expression*. El tipo *Current\_item* depende del tipo *Expression*. El bucle `For each... End for each` puede iterar a través de tres tipos de *Expression*:

- colecciones: bucle en cada elemento de la colección,
- selecciones de entidades: bucle en cada entidad,
- objetos: bucle en cada propiedad del objeto.

La siguiente tabla compara los tres tipos de `For each... End for each`:

	Bucle en las colecciones	Bucle en las selecciones de entidades	Bucle en los objetos
Tipo Current_Item	Variable del mismo tipo que los elementos de la colección	Entity	Variable texto
Tipos de expresiones	Colección (con elementos del mismo tipo)	Entity selection	Objeto
Número de bucles (por defecto)	Número de elementos de la colección	Número de entidades en la selección	Número de propiedades del objeto
Soporte de parámetros begin / end	Sí	Sí	No

- El número de bucles se evalúa al inicio y no cambiará durante el proceso. La adición o eliminación de elementos durante el bucle no suele ser recomendable, ya que puede resultar en redundancia o perdidas de iteraciones.
- Por defecto, la(s) *instrucciones* adjuntas se ejecutan para cada valor de *Expresión*. Sin embargo, es posible salir del bucle comprobando una condición al principio del bucle (`While`) o al final del bucle (`Until`).
- Los parámetros opcionales *begin* y *end* pueden utilizarse con colecciones y selecciones de entidades para definir los límites del bucle.

- El bucle `For each... End for each` puede utilizarse en una colección compartida o en un objeto compartido. Si su código necesita modificar uno o más elementos de la colección o de las propiedades del objeto, debe utilizar las palabras clave `Use...End use`. Dependiendo de sus necesidades, puede llamar a las palabras clave `Use...End use`:
  - antes de entrar en el bucle, si los elementos deben modificarse juntos por razones de integridad, o
  - dentro del bucle cuando sólo hay que modificar algunos elementos/propiedades y no es necesario gestionar la integridad.

Las instrucciones `break` y `continue` se [describen a continuación](#).

## Bucle en las colecciones

Cuando `For each...End for each` se utiliza con una *Expression* del tipo *Collection*, el parámetro *Current\_Item* es una variable del mismo tipo que los elementos de la colección. Si algún elemento de la colección no es del mismo tipo que la variable, se genera un error y el bucle se detiene.

La colección debe contener sólo elementos del mismo tipo, de lo contrario se devolverá un error en cuanto a la variable *Current\_Item* se le asigne el primer tipo de valor diferente.

En cada iteración del bucle, la variable *Current\_Item* se llena automáticamente con el elemento correspondiente de la colección. Hay que tener en cuenta los siguientes puntos:

- Si la variable *Current\_Item* es de tipo objeto o de tipo colección (es decir, si *Expresión* es una colección de objetos o de colecciones), al modificar esta variable se modificará automáticamente el elemento coincidente de la colección (porque los objetos y las colecciones comparten las mismas referencias). Si la variable es de tipo escalar, sólo se modificará la variable.
- La variable *Current\_Item* debe ser del mismo tipo que los elementos de la colección. Si algún elemento de la colección no es del mismo tipo que la variable, se genera un error y el bucle se detiene.
- Si la colección contiene elementos con un valor Null, se generará un error si el tipo de variable *Current\_Item* no soporta valores Null (como las variables de tipo entero largo).

## Ejemplo

Usted quiere calcular algunas estadísticas para una colección de números:

```

C_COLLECTION($nums)
$nums:=New collection(10;5001;6665;33;1;42;7850)
C_LONGINT($item;$vEven;$vOdd;$vUnder;$vOver)
For each($item;$nums)
  If($item%2=0)
    $vEven:=$vEven+1
  Else
    $vOdd:=$vOdd+1
  End if
  Case of
    :($item<5000)
      $vUnder:=$vUnder+1
    :($item>6000)
      $vOver:=$vOver+1
  End case
End for each
//$vEven=3, $vOdd=4
//$vUnder=4,$vOver=2

```

## Bucle en las selecciones de entidades

Cuando `For each... End for each` se utiliza con una *Expression* del tipo *Collection*, el parámetro *Current\_Item* es una variable del mismo tipo que los elementos de la colección.

El número de bucles se basa en el número de entidades de la selección de entidades. En cada iteración del bucle, el parámetro *Current\_Item* se llena automáticamente con la entidad de la selección de entidades que se procesa

actualmente.

Nota: si la selección de entidades contiene una entidad que fue eliminada mientras tanto por otro proceso, se salta automáticamente durante el bucle.

Tenga en cuenta que cualquier modificación aplicada en la entidad actual debe ser guardada explícitamente utilizando `entity.save()`.

## Ejemplo

Quiere aumentar el salario de todos los empleados británicos en una selección de entidades:

```
C_OBJECT(emp)
For each(emp;ds.Employees.query("country='UK'"))
    emp.salary:=emp.salary*1,03
    emp.save()
End for each
```

## Bucles en las propiedades de objetos

Cuando se utiliza `For each...` `End for each` con una *Expression* de tipo Object, el parámetro *Current\_Item* es una variable texto que se llena automáticamente con el nombre de la propiedad actualmente procesada.

Las propiedades del objeto se procesan de acuerdo con su orden de creación. Durante el bucle, se pueden añadir o eliminar propiedades en el objeto, sin modificar el número de bucles que quedarán en función del número original de propiedades del objeto.

## Ejemplo

Quiere pasar los nombres a mayúsculas en el siguiente objeto:

```
{
    "firstname": "gregory",
    "lastname": "badikora",
    "age": 20
}
```

Puede escribir:

```
For each(property;vObject)
    If(Value_type(vObject[property])=Is text)
        vObject[property]:=Uppercase(vObject[property])
    End if
End for each
```

```
{
    "firstname": "GREGORY",
    "lastname": "BADIKORA",
    "age": 20
}
```

## Parámetros begin / end

Puede definir los límites de la iteración utilizando los parámetros opcionales inicio y fin.

Nota: los parámetros *inicio* y *fin* sólo pueden utilizarse en iteraciones a través de colecciones y selecciones de entidades (se ignoran en las propiedades de objetos).

- En el parámetro `begin`, pase la posición del elemento en `Expression` en la que se iniciará la iteración (se incluye `begin`).
- En el parámetro `end`, también se puede pasar la posición del elemento en `Expression` en la que se debe detener la iteración (se excluye `end`).

Si se omite `fin` o si `fines` mayor que el número de elementos de `Expression`, se iteran los elementos desde `inicio` hasta el último (incluido). Si los parámetros `inicio` y `fin` son valores positivos, representan posiciones reales de elementos en `Expression`. Si `comienzo` es un valor negativo, se recalcula como `comienzo:=comienzo+tamaño de la expresión` (se considera como el desplazamiento desde el final de `Expression`). Si el valor calculado es negativo, `inicio` toma el valor 0. Nota: aunque `inicio` sea negativo, la iteración se sigue realizando en el orden estándar. Si `fin` es un valor negativo, se recalcula como `fin:=fin+tamaño de la expresión`

Por ejemplo:

- una colección contiene 10 elementos (numerados de 0 a 9)
- `begin=-4 > -> begin=-4+10=6 >->` la iteración comienza en el sexto elemento (#5)
- `end=-2 -> end=-2+10=8 ->` la iteración se detiene antes del 8º elemento (#7), es decir, en el 7º elemento.

## Ejemplo

```
C_COLLECTION($col;$col2)
$col:=New collection("a";"b";"c";"d";"e")
$col2:=New collection(1;2;3)
C_TEXT($item)
For each($item;$col;0;3)
    $col2.push($item)
End for each
//$col2=[1,2,3,"a","b","c"]
For each($item;$col;-2;-1)
    $col2.push($item)
End for each
//$col2=[1,2,3,"a","b","c","d"]
```

## Condiciones Until y While

Puede controlar la ejecución de `For each... End for each` añadiendo una condición `Until` o una condición `While` al bucle. Cuando una instrucción `Until(condición)` está asociada al bucle, la iteración se detendrá tan pronto como la condición se evalúe como `True`, mientras que cuando se trata de una instrucción `While(condición)`, la iteración se detendrá cuando la condición se evalúe por primera vez como `False`.

Puede pasar cualquiera de las dos palabras clave en función de sus necesidades:

- La condición `Until` se comprueba al final de cada iteración, por lo que si `Expression` no está vacía o es nula, el bucle se ejecutará al menos una vez.
- La condición `While` se prueba al principio de cada iteración, por lo que según el resultado de la condición, el bucle puede no ejecutarse en absoluto.

## Ejemplo

```

$colNum:=New collection(1;2;3;4;5;6;7;8;9;10)

$total:=0
For each($num;$colNum)While($total<30) //probada al inicio
    $total:=$total+$num
End for each
ALERT(String($total)) //$/total = 36 (1+2+3+4+5+6+7+8)

$total:=1000
For each($num;$colNum)Until($total>30) //probada al final
    $total:=$total+$num
End for each
ALERT(String($total)) //$/total = 1001 (1000+1)

```

## break y continue

Todas las estructuras de bucle anteriores soportan las instrucciones `break` y `continue`. Estas instrucciones le dan más control sobre los bucles al permitir salir del bucle y pasar por alto la iteración actual en cualquier momento.

### break

La instrucción `break` termina el bucle que la contiene. El control del programa fluye hacia la instrucción inmediatamente posterior al cuerpo del bucle.

Si la instrucción `break` está dentro de un bucle al interior de un [bucle anidado](#) (bucle dentro de otro bucle), la instrucción `break` terminará el bucle más interno.

### Ejemplo

```

For (vCounter;1;100)
    If ($tab{vCounter}!="") //si una condición es verdadera
        break //end of the for loop
    End if
End for

```

### continue

La instrucción `continue` termina la ejecución de las instrucciones en la iteración actual del bucle actual, y continúa la ejecución del bucle con la siguiente iteración.

```

var $text : Text
For ($i; 0; 9)
    If ($i=3)
        continue //pasar directamente a la siguiente iteración
    End if
    $text:=$text+String($i)
End for
// $text="012456789"

```

# Gestión de errores

El manejo de errores es el proceso de anticipar y responder a los errores que puedan ocurrir en su aplicación. 4D soporta de forma completa la detección y notificación de errores en tiempo de ejecución, así como el análisis de sus condiciones.

La gestión de errores responde a dos necesidades principales:

- descubrir y corregir posibles errores y fallos en el código durante la fase de desarrollo,
- detectar y recuperar errores inesperados en las aplicaciones desplegadas; en particular, puede sustituir los diálogos de error del sistema (disco lleno, archivo perdido, etc.) por su propia interfaz.

Es muy recomendable instalar un método de gestión de errores en 4D Server, para todo el código que se ejecute en el servidor. Este método evitaría la aparición de cajas de diálogo inesperadas en el servidor, y podría registrar los errores en un archivo específico para su posterior análisis.

## Error o estado

Muchas funciones de clase 4D, como `entity.save()` o `transporter.send()`, devuelven un objeto `status`. Este objeto se utiliza para almacenar errores "predecibles" en el contexto de ejecución, por ejemplo, una contraseña no válida, una entidad bloqueada, etc., que no detienen la ejecución del programa. Esta categoría de errores puede ser manejada por el código habitual.

Otros errores "imprevisibles" son el error de escritura en el disco, el fallo de la red o, en general, cualquier interrupción inesperada. Esta categoría de errores genera excepciones y necesita ser manejada a través de un método de gestión de errores.

## Instalación de un método de gestión de errores

En 4D, todos los errores pueden ser captados y manejados en un método proyecto específico, el método `gestión de errores` (o `captura de errores`).

Este método proyecto se instala para el proceso actual y será llamado automáticamente para cualquier error que se produzca en el proceso, en modo interpretado o compilado. Para *instalar* este método proyecto, basta con llamar al comando `ON ERR CALL` con el nombre del método proyecto como parámetro. Por ejemplo:

```
ON ERR CALL("IO_ERRORS") //Instala el método de gestión de errores
```

Para dejar de detectar errores y devolver el control a 4D, llame a `ON ERR CALL` con una cadena vacía:

```
ON ERR CALL("") //devuelve el control a 4D
```

El comando `Method called on error` le permite conocer el nombre del método instalado por `ON ERR CALL` para el proceso actual. Es particularmente útil en el contexto de código genérico porque permite cambiar temporalmente y luego restaurar el método de captura de error:

```

$methCurrent:=Method called on error
ON ERR CALL("NewMethod")
//Si no se puede abrir el documento, se genera un error
$ref:=Open document("MyDocument")
//Reinstalación del método anterior
ON ERR CALL($methCurrent)

```

## Alcance y componentes

Se puede definir un único método de captura de errores para toda la aplicación o diferentes métodos por módulo de aplicación. Sin embargo, sólo se puede instalar un método por proceso.

Un método de gestión de errores instalado por el comando `ON ERR CALL` sólo se aplica únicamente a la aplicación en ejecución. En el caso de un error generado por un componente, no se llama al método `ON ERR CALL` de gestión de errores de la aplicación local, y viceversa.

## Manejo de errores dentro del método

Dentro del método de error personalizado, tiene acceso a varias piezas de información que le ayudarán a identificar el error:

- variables sistema (\*):
  - `Error` (entero largo): código de error
  - `Error method` (texto): nombre del método que ha provocado el error
  - `Error line` (entero largo): número de línea del método que ha provocado el error
  - `Error formula` (texto): fórmula del código 4D (texto bruto) que está en el origen del error.

(\*) 4D mantiene automáticamente una serie de variables llamadas variables sistema, que responden a diferentes necesidades. Consulte el *Manual del lenguaje de 4D*.

- el comando `GET LAST ERROR STACK` que devuelve información sobre la pila de errores actual de la aplicación 4D.
- el comando `Get call chain` que devuelve una colección de objetos que describen cada paso de la cadena de llamadas a métodos dentro del proceso actual.

## Ejemplo

He aquí un sistema de gestión de errores sencillo:

```

//instalar el método de gestión de errores
ON ERR CALL("errorMethod")
//... ejecutar el código
ON ERR CALL("") //giving control back to 4D

```

```

// método proyecto errorMethod
If(Error#1006) //esto no es una interrupción del usuario
  ALERT("Se ha producido el error "+String(Error)+". El código en cuestión es: \\""+Error formula+"\\"")
End if

```

## Utilizar un método de gestión de errores vacío

Si desea principalmente que la caja de diálogo de error estándar esté oculta, puede instalar un método de gestión de errores vacío. La variable sistema `Error` puede ser probada en cualquier método, es decir, fuera del método de gestión de errores:

```
ON ERR CALL("emptyMethod") //emptyMethod existe pero está vacío
$doc:=Open document( "myFile.txt")
If (Error=-43)
    ALERT("File not found.")
End if
ON ERR CALL("")
End if
ON ERR CALL("")
End if
ON ERR CALL("")
```

# Modos interpretado y compilado

Las aplicaciones 4D pueden funcionar en modo interpretado o compilado:

- en modo interpretado, las declaraciones se leen y se traducen en lenguaje máquina en el momento de su ejecución. Puede añadir o modificar el código siempre que lo necesite, la aplicación se actualiza automáticamente.
- en modo compilado, todos los métodos se leen y traducen una vez, en el paso de compilación. Después, la aplicación sólo contiene instrucciones a nivel de ensamblaje, ya no es posible editar el código.

Las ventajas de la compilación son:

- Velocidad: su base aplicación se ejecuta de 3 a 1.000 veces más rápido.
- Verificación del código: su aplicación se analiza para comprobar la consistencia del código. Se detectan tanto los conflictos lógicos como los sintácticos.
- Protección:: una vez compilada su aplicación, puede eliminar el código interpretado. Entonces, la aplicación compilada es funcionalmente idéntica a la original, excepto que la estructura y los métodos no pueden ser vistos o modificados, deliberada o inadvertidamente.
- Aplicaciones independientes con doble clic: las aplicaciones compiladas también pueden transformarse en aplicaciones independientes (archivos.EXE) con su propio ícono.
- Modo apropiativo: sólo se puede ejecutar código compilado en procesos apropiativos.

## Diferencias entre el código interpretado y el compilado

Aunque la aplicación funcionará de la misma manera en modo interpretado y compilado, hay algunas diferencias que hay que conocer cuando se escribe código que será compilado. El intérprete de 4D suele ser más flexible que el compilador.

Compilado	Interpretado
No se puede tener un método con el mismo nombre que una variable.	No se genera ningún error, pero se da prioridad al método
Todas las variables deben ser tipificadas, ya sea a través de una directiva del compilador (ej.: <code>C_LONGINT</code> ), o por el compilador en tiempo de compilación.	Las variables se pueden escribir sobre la marcha (no se recomienda)
No se puede cambiar el tipo de datos de ninguna variable o array.	Es posible cambiar el tipo de datos de una variable o un array (no se recomienda)
No se puede cambiar un array unidimensional a uno bidimensional, ni cambiar un array bidimensional a uno unidimensional.	Possible
Aunque el compilador escribirá la variable por usted, debe especificar el tipo de datos de una variable utilizando directivas del compilador cuando el tipo de datos sea ambiguo, como en un formulario.	
La función <code>Undefined</code> siempre devuelve <code>False</code> para las variables. Las variables siempre están definidas.	
Si has marcado la propiedad "Puede ser ejecutado en procesos apropiativos" para el método, el código no debe llamar a ningún comando hilo no seguro u otros métodos hilo no seguro.	Se ignoran las propiedades de los procesos preventivos
El comando <code>IDLE</code> es necesario para llamar a 4D en bucles específicos	Siempre es posible interrumpir 4D

## Utilizar las directivas del compilador con el intérprete

Las aplicaciones no compiladas no requieren directivas del compilador. El intérprete digita automáticamente cada

variable en función de cómo se utilice en cada declaración, y una variable puede volver a escribirse libremente en el proyecto de aplicación.

Debido a esta flexibilidad, es posible que una aplicación tenga un rendimiento diferente en modo interpretado y compilado.

Por ejemplo, si se escribe:

```
C_LONGINT(MyInt)
```

y en otra parte del proyecto, escribe:

```
MyInt:=3.1416
```

En este ejemplo, `MyInt` se asigna el mismo valor (3) tanto en el modo interpretado como en el compilado, siempre que la directiva del compilador se interprete *antes* de la declaración de asignación.

El intérprete 4D utiliza directivas de compilador para escribir las variables. Cuando el intérprete encuentra una directiva de compilador, escribe la variable según la directiva. Si una declaración posterior intenta asignar un valor incorrecto (por ejemplo, asignar un valor alfanumérico a una variable numérica) la asignación no tendrá lugar y generará un error.

El orden en el que aparecen las dos declaraciones es irrelevante para el compilador, porque primero busca en todo el proyecto para las directivas del compilador. El intérprete, sin embargo, no es sistemático. Interpreta las declaraciones en el orden de ejecución. Ese orden, por supuesto, puede cambiar de una sesión a otra, dependiendo de lo que haga el usuario. Por esta razón, es importante diseñar su proyecto de manera que las directivas del compilador se ejecuten antes de cualquier declaración que contenga las variables declaradas.

## Utilización de punteros para evitar la reescritura

Una variable no se puede volver a escribir. Sin embargo, es posible utilizar un puntero para referirse a variables de diferentes tipos de datos. Por ejemplo, el siguiente código está permitido tanto en modo interpretado como compilado:

```
C_POINTER($p)
C_TEXT($name)
C_LONGINT($age)

$name:="Smith"
$age:=50

$p:==>$name //texto objetivo para el puntero
$p->:="Wesson" //asigna un valor texto

$p:==>$age
// nuevo objetivo de tipo diferente para el puntero
$p->:=55 //asigna un valor numérico
```

Imagine una función que devuelve la longitud (número de caracteres) de valores de todo tipo.

```
// Calc_Length (cuántos caracteres)
// $1 = puntero a tipo de variable flexible, numérica, texto, hora, booleana

C_POINTER($1)
C_TEXT($result)
C_LONGINT($0)
$result:=String($1->)
$0:=Length($result)
```

Entonces se puede llamar a este método:

```
$var1:="my text"
$var2:=5.3
$var3:=?10:02:24?
$var1:="my text"
$var2:=5.3
$var3:=?10:02:24?
$var1:="my text"
$var2:=5.3
$var3:=?10:02:24?
$var1:="my text"
$var2:=5.3
$var3:=?10:02:24?
$var4:=True

$vLength:=Calc_Length(->$var1)+Calc_Length(->$var2)+Calc_Length (->$var3)+Calc_Length(->$var4)

ALERT("Total length: "+String($vLength))
```

# Componentes

Un componente 4D es un conjunto de métodos y formularios 4D que representan una o varias funcionalidades que pueden instalarse y utilizarse en sus proyectos. Por ejemplo, el [componente 4D SVG](#) añade comandos avanzados y un motor de renderizado integrado que puede utilizarse para mostrar los archivos SVG.

## ¿Dónde encontrar los componentes?

Varios componentes están [preinstalados en el entorno de desarrollo 4D](#), pero muchos componentes 4D de la comunidad 4D [están disponibles en GitHub](#). Adicionalmente, puede [desarrollar sus propios componentes 4D](#).

## Instalación de los componentes

To install a component, you simply need to copy the component files into the [Components folder of the project](#). Puede utilizar alias o atajos.

Un proyecto local que se ejecuta en modo interpretado puede utilizar componentes interpretados o compilados. Un proyecto local que se ejecuta en modo compilado no puede utilizar componentes interpretados. En este caso, sólo se pueden utilizar componentes compilados.

## Utilización de los componentes

Los métodos y formularios de los componentes pueden utilizarse como elementos estándar en su desarrollo 4D.

Cuando un componente instalado contiene métodos, éstos aparecen en el tema [Métodos componente](#) de la página [Métodos del Explorador](#).

Puede seleccionar un método componente y hacer clic en el botón [Documentación del Explorador](#) para obtener información sobre el mismo, [si la hay](#).

The screenshot shows the 4D Explorer interface with the title bar "myProject - Explorateur". The left sidebar has categories: Démarrage, Tables, Formulaires, Méthodes (selected), Commandes, Constantes, Plug-ins, and Corbeille. The main pane displays the "Méthodes" section for the PushNotification component. The tree view shows nodes like Méthodes projet, Méthodes composant, 4D Labels, 4D Mobile App, 4D Mobile App Server, Action, Authentication, Dev, dev UpdateStructure, Mobile App Action, Mobile App Active Sessi..., Mobile App Authenti..., Mobile App Email Check..., MobileAppServer, PushNotification (which is selected and highlighted in blue), Session, WebHandler, 4D OAuth2, 4D Progress, 4D Report, 4D SVG, 4D ViewPro, and 4D Widgets. To the right of the tree view, there is a detailed documentation panel for the PushNotification component. It features a large bold title "PushNotification" with a yellow bell icon. Below the title is a brief description: "Utility class to send a push notification to one or multiple recipients." A section titled "Usage" provides instructions: "In order to use the component to send push notification, it is required to have an authentication key file `AuthKey_XXXXXX.p8` from Apple." It also links to "Check how to generate your authentication key .p8 file". Further down, it says: "To experiment the default behaviour, this file should be placed in your application sessions folder (`MobileApps/TEAM123456.com.sample.myappname`).". At the bottom of the documentation panel, there is a code block showing examples:

```
$pushNotification:=MobileAppServer .PushNotification.new()  
  
$notification:=New object  
$notification.title:="This is title"
```



# Plug-ins

A medida que desarrolle una aplicación 4D, descubrirá muchas funcionalidades de las que no se percató cuando empezó. Incluso puede extender la versión estándar de 4D añadiendo plug-ins a su entorno de desarrollo 4D.

## ¿Qué es un plug-in y qué puede hacer?

A plug-in is a piece of code, written in any language such as C or C++, that 4D launches at start up. Añade funcionalidad a 4D y aumenta así su capacidad. A plug-in usually contains a set of routines given to the 4D developer. It can handle external areas and run external processes.

## ¿Dónde encontrar los plug-ins?

Multiple plug-ins have already been written by the 4D community. Published plug-ins [can be found on GitHub](#). Additionnally, you can [develop your own plug-ins](#).

## Instalar un plug-in

You install plug-ins in the 4D environment by copying their files into the Plugins folder, at the [same level as the Project folder](#).

Los plug-ins son cargados por 4D cuando se lanza la aplicación, por lo que tendrá que salir de su aplicación 4D antes de instalarlos. If a plug-in requires a specific license for use, it will be loaded but not available for use.

## Utilización de los plug-ins

Plug-ins commands can be used as regular 4D commands in your 4D development. Plug-in commands appear in the Plug-ins page of the Explorer.

# Identificadores

Esta sección describe las convenciones y reglas para nombrar los distintos elementos del lenguaje 4D (variables, propiedades objeto, tablas, formularios, etc.).

Si se utilizan caracteres no romanos en los nombres de los identificadores, su longitud máxima puede ser menor.

## Arrays

Los nombres de los arrays siguen las mismas reglas que las [variables](#).

## Clases

El nombre de una clase puede contener hasta 31 caracteres.

Un nombre de clase debe cumplir con el estándar [reglas de nomenclatura de propiedades](#) para la notación de puntos.

No se recomienda dar el mismo nombre a una clase y a una [tabla de la base](#), para evitar conflictos.

## Funciones

Los nombres de función deben cumplir con el estándar [reglas de nomenclatura de propiedades](#) para la notación de puntos.

Consejo: comenzar el nombre de la función con un carácter de subrayado ("\_") excluirá la función de las funcionalidades de autocompletado en el editor de código 4D.

## Propiedades de los objetos

El nombre de una propiedad objeto (también llamado objeto *atributo*) puede contener hasta 255 caracteres.

Las propiedades de los objetos pueden hacer referencia a valores escalares, elementos ORDA, funciones de clase, otros objetos, etc. Sea cual sea su naturaleza, los nombres de las propiedades de los objetos deben seguir las siguientes reglas si se quiere utilizar la [notación de punto](#):

- El nombre de una propiedad debe comenzar con una letra, un guión bajo o un dólar "\$".
- A partir de ahí, el nombre puede incluir cualquier letra, dígito, el carácter de subrayado ("\_"), o el carácter de dólar ("\$").
- Los nombres de propiedades son sensibles a las mayúsculas y minúsculas.

Ejemplos:

```
myObject.myAttribute:="10"  
$value:=$clientObj.data.address.city
```

Si utiliza notación de cadena entre corchetes, los nombres de las propiedades pueden contener cualquier carácter (ej.: `miObjeto["1. First property"]` ).

Ver también [ECMA Script standard](#).

## Parámetros

Los nombres de los parámetros deben comenzar con un carácter `$` y seguir las mismas reglas que los [nombres de variables](#).

Ejemplos:

```
Function getArea($width : Integer; $height : Integer) -> $area : Integer  
#DECLARE ($i : Integer ; $param : Date) -> $myResult : Object
```

## Métodos proyecto

El nombre de un método proyecto puede contener hasta 31 caracteres.

- Un nombre de método proyecto debe comenzar por una letra, un dígito o un guión bajo
- A partir de ahí, el nombre puede incluir cualquier letra o dígito, el carácter de subrayado ("\_"), o el carácter de espacio.
- No utilice nombres reservados, es decir, nombres de comandos 4D (`Date`, `Time`, etc), keywords (`If`, `For`, etc.), o nombres de constantes (`Euro`, `Black`, `Friday`, etc.).
- Los nombres de métodos proyecto son sensibles a las mayúsculas y minúsculas.

Ejemplos:

```
If(New client)  
DELETE DUPLICATED VALUES  
APPLY TO SELECTION([Employees];INCREASE SALARIES)
```

Consejo: es una buena técnica de programación adoptar la misma convención de nomenclatura que la utilizada por 4D para los métodos integrados. Utilice mayúsculas para nombrar sus métodos; sin embargo, si un método devuelve un valor, ponga en mayúscula el primer carácter de su nombre. De este modo, cuando vuelva a abrir un proyecto para su mantenimiento después de unos meses, ya sabrá si un método devuelve un resultado simplemente mirando su nombre en la ventana del Explorador.

Cuando se llama a un método, sólo hay que escribir su nombre. Sin embargo, algunos comandos integrados en 4D, como `ON EVENT CALL`, así como también todos los comandos del plug-in, esperan el nombre de un método como una cadena cuando se pasa un parámetro de tipo método.

Ejemplos:

```
//Este comando espera un método (función) o una fórmula  
QUERY BY FORMULA([aTable];Special query)  
//Este comando espera un método (procedimiento) o una instrucción  
APPLY TO SELECTION([Employees];INCREASE SALARIES)  
//Pero este comando espera un nombre de método  
ON EVENT CALL("HANDLE EVENTS")
```

## Tablas y campos

Una tabla se designa colocando su nombre entre paréntesis: [...]. Para designar un campo hay que especificar primero la tabla a la que pertenece (el nombre del campo sigue inmediatamente al nombre de la tabla).

Un nombre de tabla y un nombre de campo pueden contener hasta 31 caracteres.

- Un nombre de tabla o campo debe comenzar con una letra, un guión bajo o un signo dólar ("\$")
- A partir de ahí, el nombre puede incluir caracteres alfabéticos, numéricos, el carácter espacio y el carácter de

- Los nombres de las cadenas pueden contener cualquier carácter.
  - Los nombres de las cadenas son sensibles a las mayúsculas y minúsculas.
- subrayado ("\_").
- No utilice nombres reservados, es decir, nombres de comandos 4D ( Date , Time , etc), keywords ( If , For , etc.), o nombres de constantes ( Euro , Black , Friday , etc.).
  - Deben respetarse reglas adicionales cuando la base deba manejarse vía SQL: sólo se aceptan los caracteres \_0123456789abcdefghijklmnopqrstuvwxyz, y el nombre no debe incluir ninguna palabra clave SQL (comando, atributo, etc.).

Ejemplos:

```
FORM SET INPUT([Clients];"Entry")
ADD RECORD([Letters])
[Orders]Total:=Sum([Line]Amount)
QUERY([Clients];[Clients]Name="Smith")
[Letters]Text:=Capitalize_text([Letters]Text)
```

No se recomienda dar el mismo nombre a una tabla y a una clase, para evitar conflictos.

## Variables

El nombre de una variable puede tener hasta 31 caracteres, sin incluir los símbolos de alcance (\$) o <>).

- Un nombre de variable debe comenzar con una letra, un guión bajo o un dólar ("\$") para parámetros y variables locales, o "<>" para variables de interproceso.
- Un dígito como primer carácter está permitido pero no se recomienda, y no es soportado por la sintaxis de declaración var .
- A partir de ahí, el nombre puede incluir cualquier letra o dígito, y el carácter de subrayado ("\_").
- Un carácter de espacio está permitido pero no se recomienda, y no es soportado por la sintaxis de declaración var .
- No utilice nombres reservados, es decir, nombres de comandos 4D ( Date , Time , etc), keywords ( If , For , etc.), o nombres de constantes ( Euro , Black , Friday , etc.).
- Los nombres de las variables son sensibles a las mayúsculas y minúsculas.

Ejemplos:

```
For($vlRecord;1;100) //variable local
$vsMyString:="Hello there" //variable local
var $vName; $vJob : Text //variables locales
If(bValidate=1) //variable proceso
<>vlProcessID:=Current process() //variable interproceso
```

## Otros nombres

En el lenguaje 4D, varios elementos tienen sus nombres manejados como cadenas: formularios, objetos de formulario, selecciones temporales, procesos, conjuntos, barras de menú, etc.

Estos nombres de cadena pueden contener hasta 255 caracteres, sin incluir los caracteres "\$" o "<>" (si los hay).

- Los nombres de las cadenas pueden contener cualquier carácter.
- Los nombres de las cadenas son sensibles a las mayúsculas y minúsculas.

Ejemplos:

```
DIALOG([Storage];"Note box"+String($vlStage))
OBJECT SET FONT(*;"Binfo";"Times")
USE NAMED SELECTION([Customers];"Closed")//Process Named Selection
USE NAMED SELECTION([Customers];"<>ByZipcode") //Interprocess Named Selection
    //Starting the global process "Add Customers"
$vlProcessID:=New process("P_ADD_CUSTOMERS";48*1024;"Add Customers")
    //Starting the local process "$Follow Mouse Moves"
$vlProcessID:=New process("P_MOUSE_SNIFFER";16*1024;"$Follow Mouse Moves")
CREATE SET([Customers];"Customer Orders")//Process set
USE SET("<>Deleted Records") //Interprocess set
If(Records in set("$Selection"+String($i))>0) //Client set
```

# Objeto del modelo de datos

The ORDA technology is based upon an automatic mapping of an underlying database structure. It also provides access to data through entity and entity selection objects. As a result, ORDA exposes the whole database as a set of data model objects.

## Cartografía de la estructura

When you call a datastore using the `ds` or the `Open datastore` command, 4D automatically references tables and fields of the corresponding 4D structure as properties of the returned `datastore` object:

- Las tablas correspondientes a las dataclasses.
- Los campos corresponden a los atributos de almacenamiento.
- Relations are mapped to relation attributes - relation names, defined in the Structure editor, are used as relation attribute names.

The screenshot shows the 4D Inspector window with the following details:

- Definition:**
  - From: [Project]companyID
  - To: [Company]ID
  - Color: Automatic
- Many to One Options:**
  - Name: **theClient** (highlighted with a red oval)
  - Manual
  - Auto Wildcard support
  - Wildcard Choice
    - ID
    - name
    - discount
  - Prompt if related one does not exist
- One to Many Options:**
  - Name: **companyProjects** (highlighted with a red oval)
  - Manual
  - Auto assign related value in subform

**Expression** table:

Expression	Value
ds.Company	DataClass: Company
ds.Company.companyProjects	{"name":"companyProjects","kind":"relatedEntities","relatedDataClass":"Project","type":"ProjectSelection"} ("name":"discount","kind":"storage","type":"number","fieldNumber":3) ("name":"ID","kind":"storage","type":"long","fieldNumber":1) ("name":"name","kind":"storage","type":"string","fieldNumber":2)
ds.Project	DataClass: Project
ds.Project.companyID	{"name":"companyID","kind":"storage","type":"long","fieldNumber":3}
ds.Project.ID	("name":"ID","kind":"storage","type":"long","fieldNumber":1)
ds.Project.name	("name":"name","kind":"storage","type":"string","fieldNumber":2)
ds.Project.theClient	("name":"theClient","kind":"relatedEntity","relatedDataClass":"Company","type":"Company")

## Reglas generales

Se aplican las siguientes reglas para todas las conversiones:

- Los nombres de tabla, campo y relación se mapean a los nombres de propiedad de objeto. Asegúrese de que dichos nombres cumplen con las reglas generales de denominación de objetos, como se explica en la sección [Convenciones de denominación de objetos](#).
- Un datastore sólo hace referencia a las tablas con una sola llave primaria. Las siguientes tablas no están referenciadas:
  - Tablas sin llave primaria
  - Tablas con llaves primarias compuestas.
- BLOB fields are automatically available as attributes of the [Blob object](#) type.

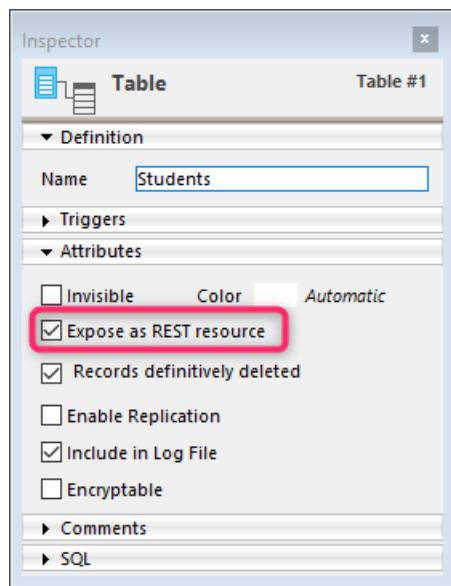
El mapeo ORDA no tiene en cuenta:

- la opción "Invisible" para tablas o campos, - la estructura virtual definida mediante `SET TABLE TITLES` o `SET FIELD TITLES`, - la propiedad "Manual" o "Automática" de las relaciones.

## Normas de control de acceso remoto

When accessing a remote datastore through the `Open datastore` command or [REST requests](#), only tables and fields with the Expose as REST resource property are available remotely.

This option must be selected at the 4D structure level for each table and each field that you want to be exposed as dataclass and attribute in the datastore:



## Actualización del modelo de datos

Any modifications applied at the level of the database structure invalidate the current ORDA model layer. Estas modificaciones incluyen:

- la adición o la eliminación de una tabla, de un campo, o de una relación
- el cambio de nombre de una tabla, de un campo o de una relación
- changing a core property of a field (type, unique, index, autoincrement, null value support)

When the current ORDA model layer has been invalidated, it is automatically reloaded and updated in subsequent calls of the local `ds` datastore on 4D and 4D Server. Note that existing references to ORDA objects such as entities or entity selections will continue to use the model from which they have been created, until they are regenerated.

However, the updated ORDA model layer is not automatically available in the following contexts:

- a remote 4D application connected to 4D Server -- the remote application must reconnect to the server.
- a remote datastore opened using `Open datastore` or through [REST calls](#) -- a new session must be opened.

## Definiciones de los objetos

### Datastore

The datastore is the interface object to a database. It builds a representation of the whole database as object. A datastore is made of a model and data:

- The model contains and describes all the dataclasses that make up the datastore. It is independant from the underlying database itself.
- Data refers to the information that is going to be used and stored in this model. For example, names, addresses, and birthdates of employees are pieces of data that you can work with in a datastore.

When handled through the code, the datastore is an object whose properties are all of the [dataclasses](#) which have been specifically exposed.

4D le permite gestionar los siguientes datastores:

- the local datastore, based on the current 4D database, returned by the `ds` command (the main datastore).
- one or more remote datastore(s) exposed as REST resources in remote 4D databases, returned by the `Open datastore` command.

A datastore references only a single local or remote database.

The datastore object itself cannot be copied as an object:

```
$mydatastore:=OB Copy(ds) //devuelve null
```

Las propiedades del datastore son sin embargo enumerables:

```
ARRAY TEXT($prop;0)
OB GET PROPERTY NAMES(ds;$prop)
//$prop contains the names of all the dataclasses
```

The main (default) datastore is always available through the `ds` command, but the `Open datastore` command allows referencing any remote datastore.

## Dataclass

Una dataclass es el equivalente de una tabla. It is used as an object model and references all fields as attributes, including relational attributes (attributes built upon relations between dataclasses). Relational attributes can be used in queries like any other attribute.

All dataclasses in a 4D project are available as a property of the `ds` datastore. For remote datastores accessed through `Open datastore` or [REST requests](#), the Expose as REST resource option must be selected at the 4D structure level for each exposed table that you want to be exposed as dataclass in the datastore.

For example, consider the following table in the 4D structure:

Company	
ID	2 <sup>32</sup>
name	A
creationDate	17/01/2018
revenues	0.5
extra	{ }

The `Company` table is automatically available as a dataclass in the `ds` datastore. Puede escribir:

```
var $compClass : cs.Company //declares a $compClass object variable of the Company class
$compClass:=ds.Company //assigns the Company dataclass reference to $compClass
```

Un objeto dataclass puede contener:

- attributes
- atributos relacionales

The dataclass offers an abstraction of the physical database and allows handling a conceptual data model. The dataclass is the only means to query the datastore. Una consulta se hace desde una única dataclass. Queries are built around attributes and relation attribute names of the dataclasses. So the relation attributes are the means to involve several linked tables in a query.

The dataclass object itself cannot be copied as an object:

```
$mydataclass:=OB Copy(ds.Employee) //devuelve null
```

Las propiedades de la dataclass son sin embargo enumerables:

```
ARRAY TEXT($prop;0)
OB GET PROPERTY NAMES(ds.Employee;$prop)
//$prop contains the names of all the dataclass attributes
```

## Atributo

Dataclass properties are attribute objects describing the underlying fields or relations. Por ejemplo:

```
$nameAttribute:=ds.Company.name //reference to class attribute
$revenuesAttribute:=ds.Company["revenues"] //alternate way
```

This code assigns to \$nameAttribute and \$revenuesAttribute references to the name and revenues attributes of the Company class. Esta sintaxis NO devuelve los valores mantenidos dentro del atributo, sino que devuelve referencias a los propios atributos. To handle values, you need to go through [Entities](#).

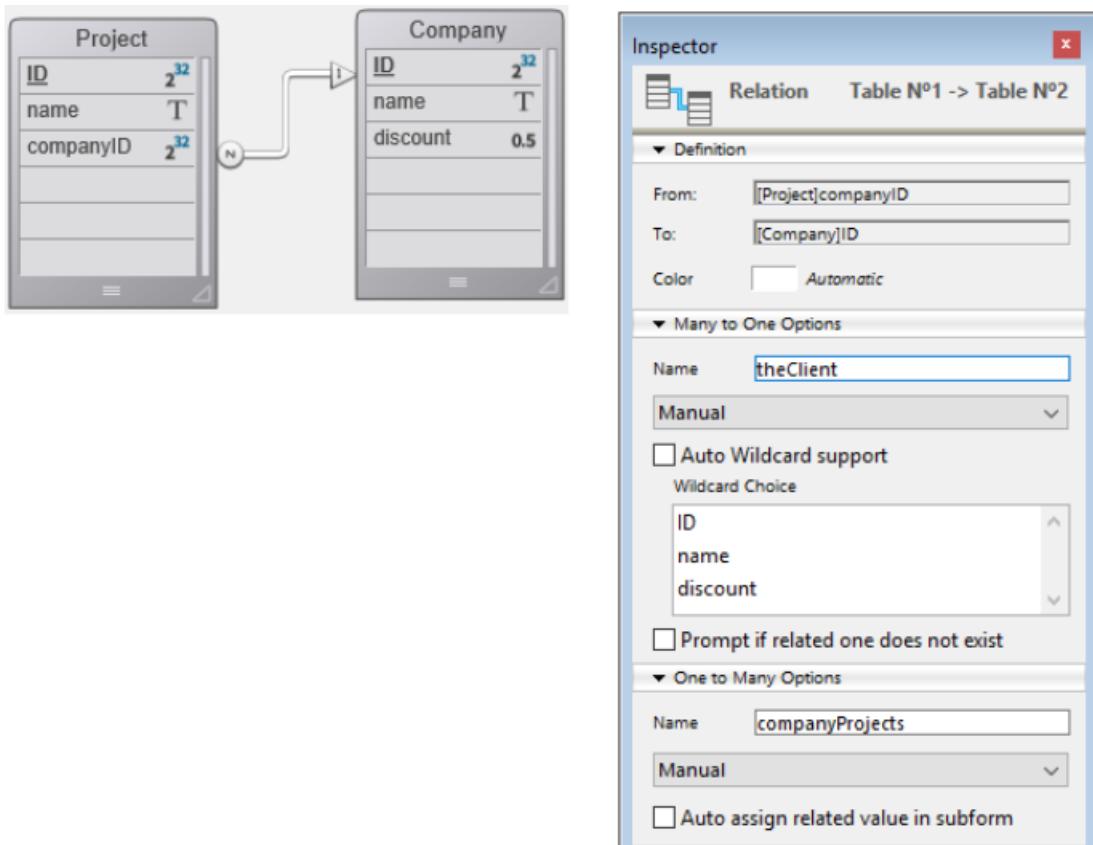
All eligible fields in a table are available as attributes of their parent [dataclass](#). For remote datastores accessed through [Open datastore](#) or [REST requests](#), the Expose as REST resource option must be selected at the 4D structure level for each field that you want to be exposed as a dataclass attribute.

### Atributos de almacenamiento y relacionales

Dataclass attributes come in several kinds: storage, relatedEntity, and relatedEntities. Attributes that are scalar (*i.e.*, provide only a single value) support all the standard 4D data types (integer, text, object, etc.).

- A storage attribute is equivalent to a field in the 4D database and can be indexed. Values assigned to a storage attribute are stored as part of the entity when it is saved. When a storage attribute is accessed, its value comes directly from the datastore. Storage attributes are the most basic building block of an entity and are defined by name and data type.
- A relation attribute provides access to other entities. Relation attributes can result in either a single entity (or no entity) or an entity selection (0 to N entities). Relation attributes are built upon "classic" relations in the relational structure to provide direct access to related entity or related entities. Relation attributes are directly available in ORDA using their names.

For example, consider the following partial database structure and the relation properties:



All storage attributes will be automatically available:

- in the Project dataclass: "ID", "name", and "companyID"
- in the Company dataclass: "ID", "name", and "discount"

In addition, the following relation attributes will also be automatically available:

- in the Project dataclass: theClient attribute, of the "relatedEntity" kind; there is at most one Company for each Project (the client)
- in the Company dataclass: companyProjects attribute, of the "relatedEntities" kind; for each Company there is any number of related Projects.

The Manual or Automatic property of a database relation has no effect in ORDA.

All dataclass attributes are exposed as properties of the dataclass:

Expression	Value
ds.Company	DataClass: Company
ds.Company.companyProjects	{"name": "companyProjects", "kind": "relatedEntities", "relatedDataClass": "Project", "type": "ProjectSelection"} {"name": "discount", "kind": "storage", "type": "number", "fieldNumber": 3} {"name": "ID", "kind": "storage", "type": "long", "fieldNumber": 1} {"name": "name", "kind": "storage", "type": "string", "fieldNumber": 2}
ds.Project	DataClass: Project
ds.Project.companyID	{"name": "companyID", "kind": "storage", "type": "long", "fieldNumber": 3} {"name": "ID", "kind": "storage", "type": "long", "fieldNumber": 1} {"name": "name", "kind": "storage", "type": "string", "fieldNumber": 2}
ds.Project.theClient	{"name": "theClient", "kind": "relatedEntity", "relatedDataClass": "Company", "type": "Company"}

Keep in mind that these objects describe attributes, but do not give access to data. Reading or writing data is done through [entity objects](#).

## Atributos calculados

[Computed attributes](#) are declared using a `get <attributeName>` function in the [Entity class definition](#). Their value is not stored but evaluated each time they are accessed. They do not belong to the underlying database structure, but are

built upon it and can be used as any attribute of the data model.

## Entity

Una entidad es el equivalente a un registro. It is actually an object that references a record in the database. It can be seen as an instance of a [dataclass](#), like a record of the table matching the dataclass. However, an entity also contains data correlated to the database related to the datastore.

The purpose of the entity is to manage data (create, update, delete). When an entity reference is obtained by means of an entity selection, it also retains information about the entity selection which allows iteration through the selection.

The entity object itself cannot be copied as an object:

```
$myentity:=OB Copy(ds.Employee.get(1)) //returns null
```

Sin embargo, las propiedades de la entidad son enumerables:

```
ARRAY TEXT($prop;0)
OB GET PROPERTY NAMES(ds.Employee.get(1);$prop)
//$prop contains the names of all the entity attributes
```

## Entity selection

An entity selection is an object containing one or more reference(s) to entities belonging to the same dataclass. It is usually created as a result of a query or returned from a relation attribute. Una entity selection puede contener 0, 1 o X entidades de la dataclass -- donde X puede representar el número total de entidades contenidas en la dataclass.

Ejemplo:

```
var $e : cs.EmployeeSelection //declares a $e object variable of the EmployeeSelection class type
$e:=ds.Employee.all() //assigns the resulting entity selection reference to the $e variable
```

Entity selections can be "sorted" or "unsorted" ([see below](#)).

Entity selections can also be "shareable" or "non-shareable", depending on [how they have been created](#).

The entity selection object itself cannot be copied as an object:

```
$myentitysel:=OB Copy(ds.Employee.all()) //returns null
```

The entity selection properties are however enumerable:

```
ARRAY TEXT($prop;0)
OB GET PROPERTY NAMES(ds.Employee.all();$prop)
//$prop contains the names of the entity selection properties
//("length", "00", "01"...)
```

## Entity selections ordenadas o no ordenadas

For optimization reasons, by default 4D ORDA usually creates unordered entity selections, except when you use the `orderBy( )` method or use specific options. In this documentation, unless specified, "entity selection" usually refers to an "unordered entity selection".

Ordered entity selections are created only when necessary or when specifically requested using options, i.e. in the following cases:

- result of an `orderBy()` on a selection (of any type) or an `orderBy()` on a dataclass
- result of the `newSelection()` method with the `dk keep ordered` option

Unordered entity selections are created in the following cases:

- result of a standard `query()` on a selection (of any type) or a `query()` on a dataclass,
- result of the `newSelection()` method without option,
- result of any of the comparison methods, whatever the input selection types: `or()`, `and()`, `minus()`.

The following entity selections are always ordered: > > \* entity selections returned by 4D Server to a remote client > \* entity selections built upon remote datastores.

-      entity selections returned by 4D Server to a remote client > \* entity selections built upon remote datastores.
- selecciones de entidades basadas en datastores remotos.

Note that when an ordered entity selection becomes an unordered entity selection, any repeated entity references are removed.

# Clases del modelo de datos

ORDA allows you to create high-level class functions above the data model. This allows you to write business-oriented code and "publish" it just like an API. Datastore, dataclasses, entity selections, and entities are all available as class objects that can contain functions.

For example, you could create a `getNextWithHigherSalary()` function in the `EmployeeEntity` class to return employees with a salary higher than the selected one. Sería tan sencillo como llamar:

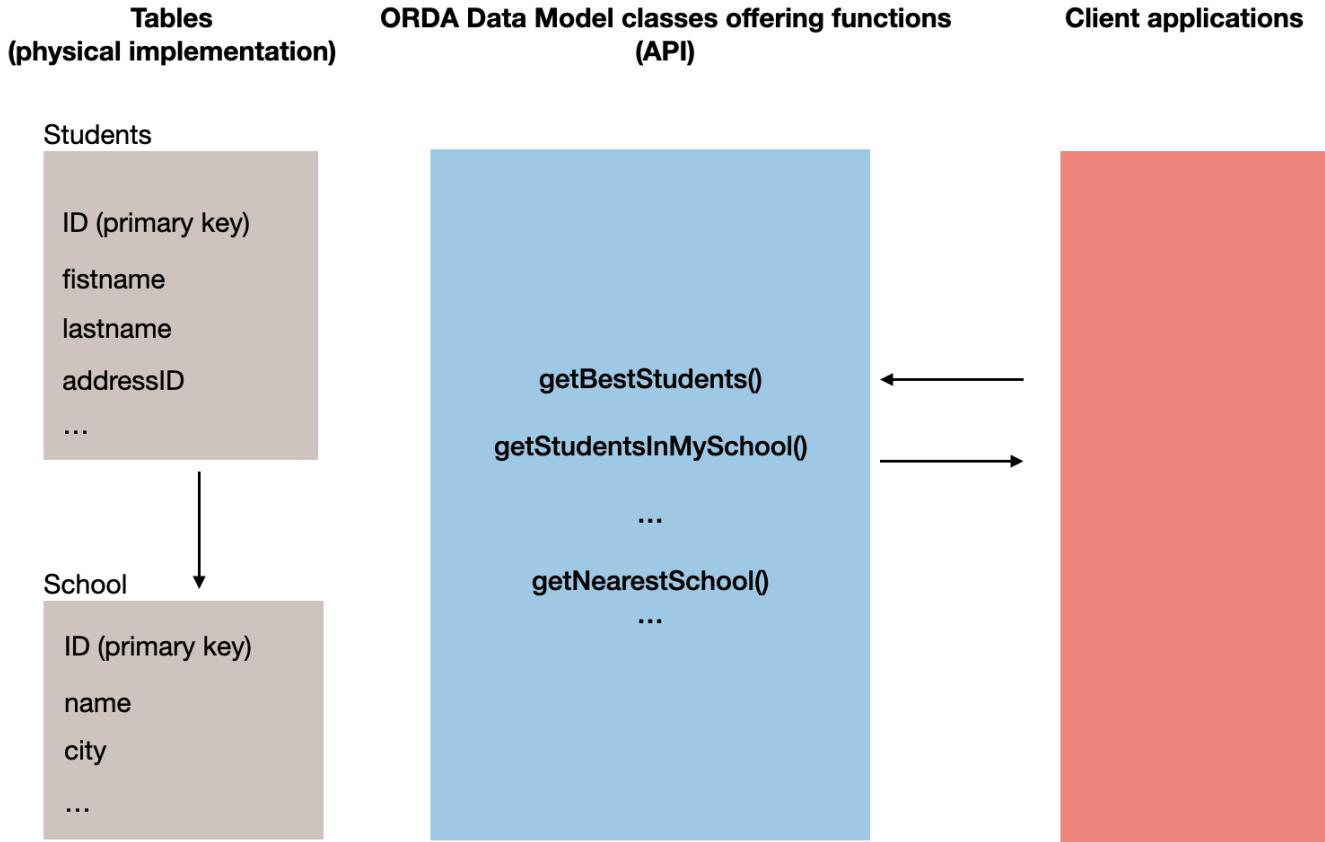
```
$nextHigh:=ds.Employee(1).getNextWithHigherSalary()
```

Developers can not only use these functions in local datastores, but also in client/server and remote architectures:

```
//$cityManager es la referencia de un datastore remoto  
Form.comp.city:=$cityManager.City.getCityName(Form.comp.zipcode)
```

Thanks to this feature, the entire business logic of your 4D application can be stored as a independent layer so that it can be easily maintained and reused with a high level of security:

- You can "hide" the overall complexity of the underlying physical structure and only expose understandable and ready-to-use functions.
- If the physical structure evolves, you can simply adapt function code and client applications will continue to call them transparently.
- By default, all of your data model class functions (including [computed attribute functions](#)) and [alias attributes](#) are not exposed to remote applications and cannot be called from REST requests. You must explicitly declare each public function and alias with the `exposed` keyword.



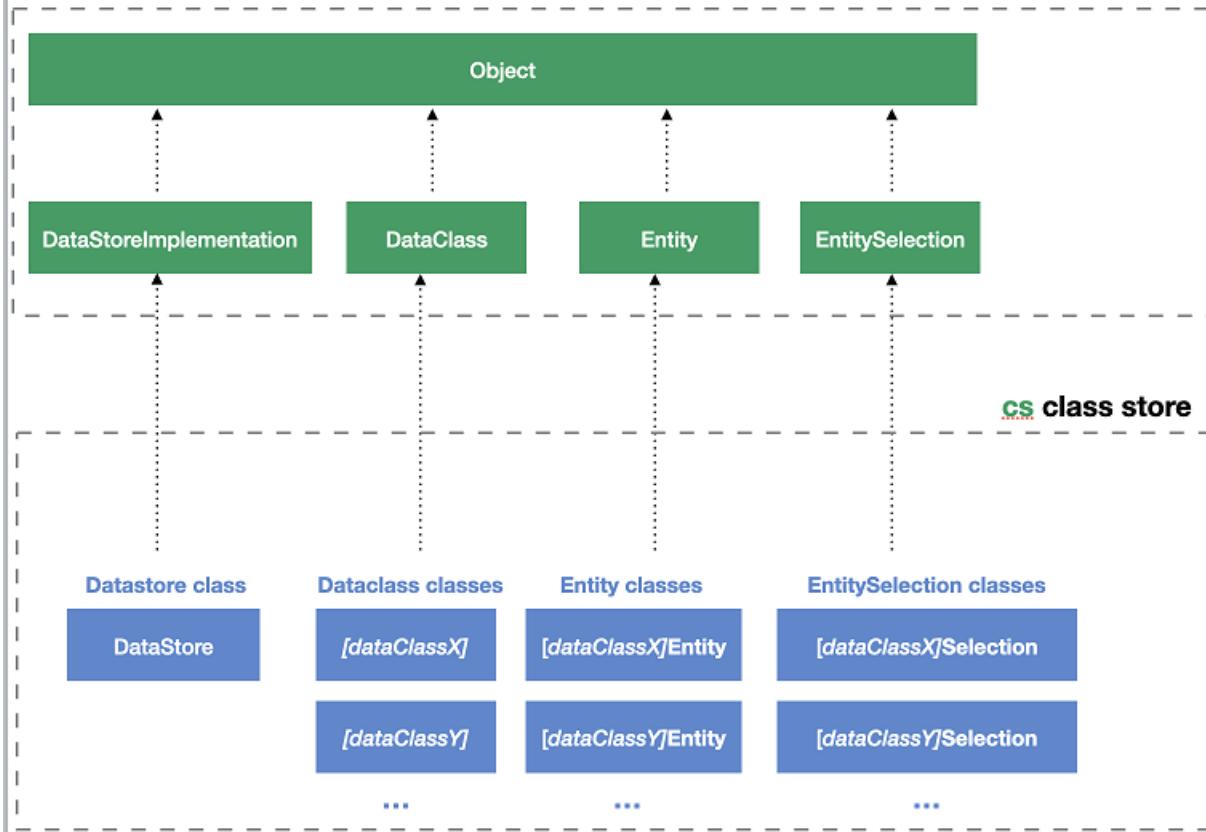
#### 4D database (eventually exposed as a REST server)

In addition, 4D [automatically pre-creates](#) the classes for each available data model object.

## Architecture

ORDA provides generic classes exposed through the `4D class store`, as well as user classes (extending generic classes) exposed in the `cs class store`:

## 4D class store



All ORDA data model classes are exposed as properties of the `cs` class store. Las clases ORDA siguientes están disponibles:

Class	Nombre del ejemplo	Instanciado por
<code>cs.DataStore</code>	<code>cs.DataStore</code>	<code>ds</code> command
<code>cs.DataClassName</code>	<code>cs.Employee</code>	<code>dataStore.DataClassName</code> , <code>dataStore["DataClassName"]</code>
<code>cs.DataClassNameEntity</code>	<code>cs.EmployeeEntity</code>	<code>dataClass.get()</code> , <code>dataClass.new()</code> , <code>entitySelection.first()</code> , <code>entitySelection.last()</code> , <code>entity.previous()</code> , <code>entity.next()</code> , <code>entity.first()</code> , <code>entity.last()</code> , <code>entity.clone()</code>
<code>cs.DataClassNameSelection</code>	<code>cs.EmployeeSelection</code>	<code>dataClass.query()</code> , <code>entitySelection.query()</code> , <code>dataClass.all()</code> , <code>dataClass.fromCollection()</code> , <code>dataClass.newSelection()</code> , <code>entitySelection.drop()</code> , <code>entity.getSelection()</code> , <code>entitySelection.and()</code> , <code>entitySelection.minus()</code> , <code>entitySelection.or()</code> , <code>entitySelection.orderBy()</code> , <code>entitySelection.orderByFormula()</code> , <code>entitySelection.slice()</code> , Create entity selection

ORDA user classes are stored as regular class files (.4dm) in the Classes subfolder of the project ([see below](#)).

Also, object instances from ORDA data model user classes benefit from their parent's properties and functions:

- a Datastore class object can call functions from the [ORDA Datastore generic class](#).
- a Dataclass class object can call functions from the [ORDA Dataclass generic class](#).
- an Entity selection class object can call functions from the [ORDA Entity selection generic class](#).
- an Entity class object can call functions from the [ORDA Entity generic class](#).

# Descripción de la clase

► Histórico

## Clase DataStore

A 4D database exposes its own DataStore class in the `cs` class store.

- Extends: 4D.DataStoreImplementation
- Nombre de clase: `cs.DataStore`

You can create functions in the DataStore class that will be available through the `ds` object.

### Ejemplo

```
// cs.DataStore class

Class extends DataStoreImplementation

Function getDesc
$0:="Database exposing employees and their companies"
```

Esta función puede ser llamada:

```
$desc:=ds.getDesc() //Database exposing..."
```

## Clase DataClass

Each table exposed with ORDA offers a DataClass class in the `cs` class store.

- Extends: 4D.DataClass
- Class name: `cs.DataClassName` (where `DataClassName` is the table name)
- Example name: `cs.Employee`

### Ejemplo

```
// cs.Company class

Class extends DataClass

// Returns companies whose revenue is over the average
// Returns an entity selection related to the Company DataClass

Function GetBestOnes()
$sel:=This.query("revenues >= :1";This.all().average("revenues"));
$0:=$sel
```

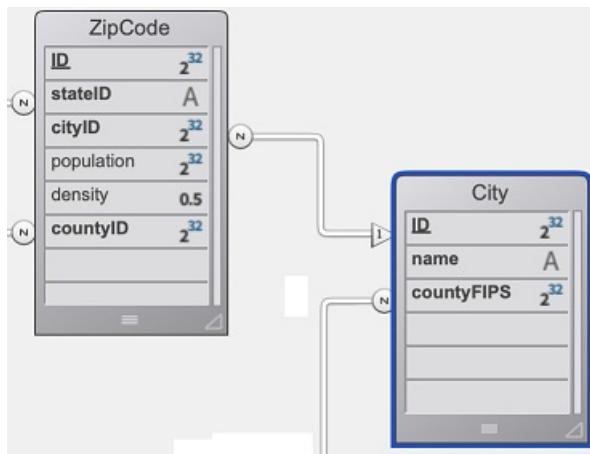
Then you can get an entity selection of the "best" companies by executing:

```
var $best : cs.CompanySelection
$best:=ds.Company.GetBestOnes()
```

Computed attributes are defined in the [Entity Class](#).

### Ejemplo con un datastore remoto

The following `City` catalog is exposed in a remote datastore (partial view):



La clase `City` ofrece una API:

```
// cs.City class

Class extends DataClass

Function getCityName()
    var $1; $zipcode : Integer
    var $zip : 4D.Entity
    var $0 : Text

    $zipcode:=$1
    $zip:=ds.ZipCode.get($zipcode)
    $0:=""

    If ($zip#Null)
        $0:=$zip.city.name
    End if
```

The client application opens a session on the remote datastore:

```
$cityManager:=Open datastore(New object("hostname";"127.0.0.1:8111");"CityManager")
```

Then a client application can use the API to get the city matching a zip code (for example) from a form:

```
Form.comp.city:=$cityManager.City.getCityName(Form.comp.zipcode)
```

## Clase EntitySelection

Each table exposed with ORDA offers an EntitySelection class in the `cs` class store.

- Extends: `4D.EntitySelection`
- Class name: `DataClassNameSelection` (where `DataClassName` is the table name)
- Example name: `cs.EmployeeSelection`

Ejemplo

```
// cs.EmployeeSelection class

Class extends EntitySelection

//Extract the employees with a salary greater than the average from this entity selection

Function withSalaryGreaterThanAverage
    C_OBJECT($0)
    $0:=This.query("salary > :1";This.average("salary")).orderBy("salary")
```

Then you can get employees with a salary greater than the average in any entity selection by executing:

```
$moreThanAvg:=ds.Company.all().employees.withSalaryGreaterThanAverage()
```

## Entity Class

Each table exposed with ORDA offers an Entity class in the `cs` class store.

- Extends: `4D.Entity`
- Class name: `DataClassNameEntity` (where `DataClassName` is the table name)
- Example name: `cs.CityEntity`

### Atributos calculados

Entity classes allow you to define computed attributes using specific keywords:

- `Function get attributeName`
- `Function set attributeName`
- `Function query attributeName`
- `Function orderBy attributeName`

For information, please refer to the [Computed attributes](#) section.

### Atributos de los alias

Entity classes allow you to define alias attributes, usually over related attributes, using the `Alias` keyword:

```
Alias attributeName targetPath
```

For information, please refer to the [Alias attributes](#) section.

## Ejemplo

```
// cs.CityEntity class

Class extends Entity

Function getPopulation()
    $0:=This.zips.sum("population")

Function isBigCity(): Boolean
// The getPopulation() function is usable inside the class
$0:=This.getPopulation()>50000
```

Luego puede llamar este código:

```

var $cityManager; $city : Object

$cityManager:=Open datastore(New object("hostname";"127.0.0.1:8111");"CityManager")
$city:=$cityManager.City.getCity("Caguas")

If ($city.isBigCity())
    ALERT($city.name + " is a big city")
End if

```

## Reglas específicas

When creating or editing data model classes, you must pay attention to the following rules:

- Since they are used to define automatic DataClass class names in the `cs class store`, 4D tables must be named in order to avoid any conflict in the cs namespace. En particular:
  - Do not give the same name to a 4D table and to a `user class name`. If such a case occurs, the constructor of the user class becomes unusable (a warning is returned by the compiler).
  - Do not use a reserved name for a 4D table (e.g., "DataClass").
- When defining a class, make sure the `Class extends` statement exactly matches the parent class name (remember that they're case sensitive). For example, `Class extends EntitySelection` for an entity selection class.
- You cannot instantiate a data model class object with the `new()` keyword (an error is returned). You must use a regular method as listed in the [Instantiated by column of the ORDA class table](#).
- You cannot override a native ORDA class function from the `4D class store` with a data model user class function.

## Ejecución apropiativa

When compiled, data model class functions are executed:

- in preemptive or cooperative processes (depending on the calling process) in single-user applications,
- in preemptive processes in client/server applications (except if the `local` keyword is used, in which case it depends on the calling process like in single-user).

If your project is designed to run in client/server, make sure your data model class function code is thread-safe. If thread-unsafe code is called, an error will be thrown at runtime (no error will be thrown at compilation time since cooperative execution is supported in single-user applications).

## Atributos calculados

### Generalidades

Un atributo calculado es un atributo de clase de datos con un tipo de datos que enmascara un cálculo. [Clases 4D estándar](#) implementa el concepto de propiedades calculadas con `get` (`getter`) y `set` (`setter`) [accessor functions](#). Los atributos de las clases de datos ORDA se benefician de esta funcionalidad y la extienden con dos funcionalidades adicionales: `query` y `orderBy`.

Como mínimo, un atributo calculado requiere una función `get` que describa cómo se calculará su valor. Cuando se suministra una función `getter` para un atributo, 4D no crea el espacio de almacenamiento subyacente en el datastore sino que sustituye el código de la función cada vez que se accede al atributo. Si no se accede al atributo, el código nunca se ejecuta.

Un atributo calculado también puede implementar una función `set`, que se ejecuta cada vez que se asigna un valor al atributo. La función `setter` describe qué hacer con el valor asignado, normalmente redirigiéndolo a uno o más atributos de almacenamiento o en algunos casos a otras entidades.

Al igual que los atributos de almacenamiento, los atributos calculados pueden incluirse en búsquedas. Por defecto, cuando se utiliza un atributo calculado en una búsqueda ORDA, el atributo se calcula una vez por entidad examinada. En algunos casos esto es suficiente. Sin embargo, para un mejor rendimiento, especialmente en cliente/servidor, los

atributos calculados pueden implementar una función `query` que se basa en los atributos reales de la clase de datos y se beneficia de sus índices.

Del mismo modo, los atributos calculados pueden incluirse en ordenaciones. Cuando se utiliza un atributo calculado en una ordenación ORDA, el atributo se calcula una vez por entidad examinada. Al igual que en las búsquedas, los atributos calculados pueden implementar una función `orderBy` que sustituya a otros atributos durante la ordenación, aumentando así el rendimiento.

## Cómo definir los atributos calculados

You create a computed attribute by defining a `get` accessor in the `entity class` of the dataclass. The computed attribute will be automatically available in the dataclass attributes and in the entity attributes.

Other computed attribute functions (`set`, `query`, and `orderBy`) can also be defined in the entity class. Son opcionales.

Within computed attribute functions, `This` designates the entity. Computed attributes can be used and handled as any dataclass attribute, i.e. they will be processed by `entity class` or `entity selection class` functions.

ORDA computed attributes are not `exposed` by default. You expose a computed attribute by adding the `exposed` keyword to the get function definition.

get and set functions can have the `local` property to optimize client/server processing.

### Function `get <attributeName>`

#### Sintaxis

```
{local} {exposed} Function get <attributeName>({$event : Object}) -> $result : type  
// code
```

The `getter` function is mandatory to declare the `attributeName` computed attribute. Whenever the `attributeName` is accessed, 4D evaluates the `Function get` code and returns the `$result` value.

A computed attribute can use the value of other computed attribute(s). Las llamadas recursivas generan errores.

The `getter` function defines the data type of the computed attribute thanks to the `$result` parameter. Se permiten los siguientes tipos resultantes:

- Scalar (text, boolean, date, time, number)
- Objeto
- Imagen
- BLOB
- Entity (i.e. cs.EmployeeEntity)
- Entity selection (p.e. cs.EmployeeeSelection)

The `$event` parameter contains the following properties:

Propiedad	Tipo	Descripción
attributeName	Texto	Nombre de atributo calculado
dataClassName	Texto	Nombre de la clase de datos
kind	Texto	"get"
result	Variant	Opcional. Add this property with Null value if you want a scalar attribute to return Null

## Ejemplos

- El campo calculado *fullName*:

```
Function get fullName($event : Object)-> $fullName : Text

Case of
: (This.firstName=NULL) & (This.lastName=NULL)
    $event.result:=Null //use result to return Null
: (This.firstName=NULL)
    $fullName:=This.lastName
: (This.lastName=NULL)
    $fullName:=This.firstName
Else
    $fullName:=This.firstName+" "+This.lastName
End case
```

- A computed attribute can be based upon an entity related attribute:

```
Function get bigBoss($event : Object)-> $result: cs.EmployeeEntity
$result:=This.manager.manager
```

- A computed attribute can be based upon an entity selection related attribute:

```
Function get coworkers($event : Object)-> $result: cs.EmployeeSelection
If (This.manager.manager=NULL)
    $result:=ds.Employee.newSelection()
Else
    $result:=This.manager.directReports.minus(this)
End if
```

## Function set <attributeName>

### Sintaxis

```
{local} Function set <attributeName>($value : type {; $event : Object})
// code
```

The *setter* function executes whenever a value is assigned to the attribute. This function usually processes the input value(s) and the result is dispatched between one or more other attributes.

The *\$value* parameter receives the value assigned to the attribute.

The *\$event* parameter contains the following properties:

Propiedad	Tipo	Descripción
attributeName	Texto	Nombre de atributo calculado
dataClassName	Texto	Nombre de la clase de datos
kind	Texto	"set"
value	Variant	Valor a tratar por el atributo calculado

## Ejemplo

```

Function set fullName($value : Text; $event : Object)
    var $p : Integer
    $p:=Position(" "; $value)
    This.firstname:=Substring($value; 1; $p-1) // "" if $p<0
    This.lastname:=Substring($value; $p+1)

```

## Function query <attributeName>

### Sintaxis

```

Function query <attributeName>($event : Object)
Function query <attributeName>($event : Object) -> $result : Text
Function query <attributeName>($event : Object) -> $result : Object
// code

```

Esta función soporta tres sintaxis:

- With the first syntax, you handle the whole query through the `$event.result` object property.
- Con la segunda y tercera sintaxis, la función devuelve un valor en `$result`:
  - If `$result` is a Text, it must be a valid query string
  - If `$result` is an Object, it must contain two properties:

Propiedad	Tipo	Descripción
<code>\$result.query</code>	Texto	Valid query string with placeholders (:1, :2, etc.)
<code>\$result.parameters</code>	Collection	valors para marcadores

The `query` function executes whenever a query using the computed attribute is launched. It is useful to customize and optimize queries by relying on indexed attributes. When the `query` function is not implemented for a computed attribute, the search is always sequential (based upon the evaluation of all values using the `get <AttributeName>` function).

The following features are not supported: - calling a `query` function on computed attributes of type Entity or Entity selection, - using the `order by` keyword in the resulting query string.

The `$event` parameter contains the following properties:

Propiedad	Tipo	Descripción
attributeName	Texto	Nombre de atributo calculado
dataClassName	Texto	Nombre de la clase de datos
kind	Texto	"query"
value	Variant	Valor a tratar por el atributo calculado
operator	Texto	Query operator (see also the <a href="#">query class function</a> ). Valores posibles: <ul style="list-style-type: none"> <li>• == (es igual a, @ es comodín)</li> <li>• === (equal to, @ is not wildcard)</li> <li>• != (no es igual a, @ es comodín)</li> <li>• !== (no es igual a, @ no es comodín)</li> <li>• &lt; (menor que)</li> <li>• &lt;= (less than or equal to)</li> <li>• &gt; (mayor que)</li> <li>• &gt;= (greater than or equal to)</li> <li>• IN (incluído en)</li> <li>• % (contiene palabra clave)</li> </ul>
result	Variant	Valor a tratar por el atributo calculado. Pass <code>Null</code> in this property if you want to let 4D execute the default query (always sequential for computed attributes).

If the function returns a value in `$result` and another value is assigned to the `$event.result` property, the priority is given to `$event.result`.

## Ejemplos

- Query on the *fullName* computed attribute.

```

Function query fullName($event : Object)->$result : Object

    var $fullname; $firstname; $lastname; $query : Text
    var $operator : Text
    var $p : Integer
    var $parameters : Collection

    $operator:=$event.operator
    $fullname:=$event.value

    $p:=Position(" "; $fullname)
    If ($p>0)
        $firstname:=Substring($fullname; 1; $p-1)+"@"
        $lastname:=Substring($fullname; $p+1)+"@"
        $parameters:=New collection($firstname; $lastname) // two items collection
    Else
        $fullname:=$fullname+"@"
        $parameters:=New collection($fullname) // single item collection
    End if

    Case of
    : ($operator=="==") | ($operator=="===")
        If ($p>0)
            $query:="(firstName = :1 and lastName = :2) or (firstName = :2 and lastName = :1)"
        Else
            $query:="firstName = :1 or lastName = :1"
        End if
    : ($operator!="!=")
        If ($p>0)
            $query:="firstName != :1 and lastName != :2 and firstName != :2 and lastName != :1"
        Else
            $query:="firstName != :1 and lastName != :1"
        End if
    End case

    $result:=New object("query"; $query; "parameters"; $parameters)

```

Keep in mind that using placeholders in queries based upon user text input is recommended for security reasons (see [query\(\) description](#)).

Código de llamada, por ejemplo:

```
$emps:=ds.Employee.query("fullName = :1"; "Flora Pionsin")
```

- This function handles queries on the `age` computed attribute and returns an object with parameters:

```

Class extends Entity

local Function age() -> $age: Variant

If (This.birthDate#!00-00-00!)
    $age:=Year of(Current date)-Year of(This.birthDate)
Else
    $age:=Null
End if

```

Código de llamada, por ejemplo:

```
// people aged between 20 and 21 years (-1 day)
$twenty:=people.query("age = 20") // calls the "==" case

// people aged 20 years today
$twentyToday:=people.query("age === 20") // equivalent to people.query("age is 20")
```

## Function orderBy <attributeName>

### Sintaxis

```
Function orderBy <attributeName>($event : Object)
Function orderBy <attributeName>($event : Object)--> $result : Text

// code
```

The `orderBy` function executes whenever the computed attribute needs to be ordered. Permite ordenar el atributo calculado. For example, you can sort `fullName` on first names then last names, or conversely. When the `orderBy` function is not implemented for a computed attribute, the sort is always sequential (based upon the evaluation of all values using the `get <AttributeName>` function).

Calling an `orderBy` function on computed attributes of type Entity class or Entity selection class is not supported.

The `$event` parameter contains the following properties:

Propiedad	Tipo	Descripción
<code>attributeName</code>	Texto	Nombre de atributo calculado
<code>dataClassName</code>	Texto	Nombre de la clase de datos
<code>kind</code>	Texto	"orderBy"
<code>value</code>	Variant	Valor a tratar por el atributo calculado
<code>operator</code>	Texto	"desc" o "asc" (por defecto)
<code>descending</code>	Booleano	<code>true</code> for descending order, <code>false</code> for ascending order
<code>result</code>	Variant	Valor a tratar por el atributo calculado. Pass <code>Null</code> if you want to let 4D execute the default sort.

You can use either the `operator` or the `descending` property. It is essentially a matter of programming style (see examples).

You can return the `orderBy` string either in the `$event.result` object property or in the `$result` function result. If the function returns a value in `$result` and another value is assigned to the `$event.result` property, the priority is given to `$event.result`.

### Ejemplo

You can write conditional code:

```

Function orderBy fullName($event : Object)-> $result : Text
  If ($event.descending=True)
    $result:="firstName desc, lastName desc"
  Else
    $result:="firstName, lastName"
  End if

```

También puede escribir un código compacto:

```

Function orderBy fullName($event : Object)-> $result : Text
  $result:="firstName "+$event.operator+", "lastName "+$event.operator

```

El código condicional es necesario en algunos casos:

```

Function orderBy age($event : Object)-> $result : Text
  If ($event.descending=True)
    $result:="birthday asc"
  Else
    $result:="birthday desc"
  End if

```

## Atributos de los alias

### Generalidades

An alias attribute is built above another attribute of the data model, named `target` attribute. The target attribute can belong to a related dataclass (available through any number of relation levels) or to the same dataclass. An alias attribute stores no data, but the path to its target attribute. You can define as many alias attributes as you want in a dataclass.

Alias attributes are particularly useful to handle N to N relations. They bring more readability and simplicity in the code and in queries by allowing to rely on business concepts instead of implementation details.

### Cómo definir los atributos alias

You create an alias attribute in a dataclass by using the `Alias` keyword in the `entity class` of the dataclass.

`Alias <attributeName> <targetPath>`

#### Sintaxis

```
{exposed} Alias <attributeName> <targetPath>
```

`attributeName` must comply with [standard rules for property names](#).

`targetPath` is an attribute path containing one or more levels, such as "employee.company.name". If the target attribute belongs to the same dataclass, `targetPath` is the attribute name.

An alias can be used as a part of a path of another alias.

A [computed attribute](#) can be used in an alias path, but only as the last level of the path, otherwise, an error is returned. For example, if "fullName" is a computed attribute, an alias with path "employee.fullName" is valid.

ORDA alias attributes are not exposed by default. You must add the `exposed` keyword before the `Alias` keyword if you want the alias to be available to remote requests.

## Uso de los atributos alias

Alias attributes are read-only (except when based upon a scalar attribute of the same dataclass, see the last example below). They can be used instead of their target attribute path in class functions such as:

Function
<code>dataClass.query()</code> , <code>entitySelection.query()</code>
<code>entity.toObject()</code>
<code>entitySelection.toCollection()</code>
<code>entitySelection.extract()</code>
<code>entitySelection.orderBy()</code>
<code>entitySelection.orderByFormula()</code>
<code>entitySelection.average()</code>
<code>entitySelection.count()</code>
<code>entitySelection.distinct()</code>
<code>entitySelection.sum()</code>
<code>entitySelection.min()</code>
<code>entitySelection.max()</code>
<code>entity.diff()</code>
<code>entity.touchedAttributes()</code>

Keep in mind that alias attributes are calculated on the server. In remote configurations, updating alias attributes in entities requires that entities are reloaded from the server.

## Alias properties

Alias attribute `kind` is "alias".

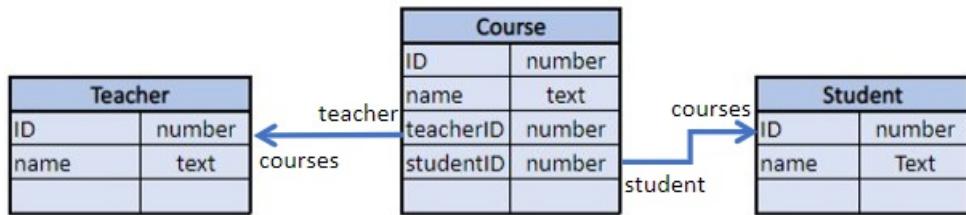
An alias attribute inherits its data `type` property from the target attribute:

- if the target attribute `kind` is "storage", the alias data type is of the same type,
- if the target attribute `kind` is "relatedEntity" or "relatedEntities", the alias data type is of the `4D.Entity` or `4D.EntitySelection` type ("classnameEntity" or "classnameSelection").

Alias attributes based upon relations have a specific `path` property, containing the path of their target attributes. Alias attributes based upon attributes of the same dataclass have the same properties as their target attributes (and no `path` property).

## Ejemplos

Considerando el siguiente modelo:



In the Teacher dataclass, an alias attribute returns all students of a teacher:

```
// cs.TeacherEntity class

Class extends Entity

Alias students courses.student //relatedEntities
```

In the Student dataclass, an alias attribute returns all teachers of a student:

```
// cs.StudentEntity class

Class extends Entity

Alias teachers courses.teacher //relatedEntities
```

En la dataclass Course:

- an alias attribute returns another label for the "name" attribute
- un atributo alias devuelve el nombre del profesor
- un atributo alias devuelve el nombre del estudiante

```
// cs.CourseEntity class

Class extends Entity

Exposed Alias courseName name //scalar
Exposed Alias teacherName teacher.name //scalar value
Exposed Alias studentName student.name //scalar value
```

Luego puede ejecutar las siguientes consultas:

```

// Find course named "Archaeology"
ds. Course.query("courseName = :1";"Archaeology")

// Find courses given by the professor Smith
ds. Course.query("teacherName = :1";"Smith")

// Find courses where Student "Martin" assists
ds. Course.query("studentName = :1";"Martin")

// Find students who have M. Smith as teacher
ds. Student.query("teachers.name = :1";"Smith")

// Find teachers who have M. Martin as Student
ds. Teacher.query("students.name = :1";"Martin")
// Note that this very simple query string processes a complex
// query including a double join, as you can see in the queryPlan:
// "Join on Table : Course : Teacher. ID = Course.teacherID,
// subquery:[ Join on Table : Student : Course.studentID = Student. ID,
// subquery:[ Student.name === Martin]]"

```

You can also edit the value of the `courseName` alias:

```

// Rename a course using its alias attribute
$arch:=ds.Course.query("courseName = :1";"Archaeology")
$arch.courseName:="Archaeology II"
$arch.save() //courseName and name are "Archaeology II"

```

## Funciones expuestas y no expuestas

For security reasons, all of your data model class functions and alias attributes are not exposed (i.e., private) by default to remote requests.

Las peticiones remotas incluyen:

- Requests sent by remote 4D applications connected through [Open datastore](#)
- Peticiones REST

Regular 4D client/server requests are not impacted. Data model class functions are always available in this architecture.

A function that is not exposed is not available on remote applications and cannot be called on any object instance from a REST request. If a remote application tries to access a non-exposed function, the "-10729 - Unknown member method" error is returned.

To allow a data model class function to be called by a remote request, you must explicitly declare it using the `exposed` keyword. La sintaxis formal es:

```
// declare an exposed function
exposed Function <functionName>
```

The `exposed` keyword can only be used with Data model class functions. If used with a [regular user class](#) function, it is ignored and an error is returned by the compiler.

## Ejemplo

You want an exposed function to use a private function in a dataclass class:

```

Class extends DataClass

//Public function
exposed Function registerNewStudent($student : Object) -> $status : Object

var $entity : cs.StudentsEntity

$entity:=ds.Students.new()
$entity.fromObject($student)
$entity.school:=This.query("name=:1"; $student.schoolName).first()
$entity.serialNumber:=This.computeSerialNumber()
$status:=$entity.save()

//Not exposed (private) function
Function computeIDNumber()-> $id : Integer
//compute a new ID number
$id:=-...

```

Cuando se llama al código:

```

var $remoteDS; $student; $status : Object
var $id : Integer

$remoteDS:=Open datastore(New object("hostname"; "127.0.0.1:8044"); "students")
$student:=New object("firstname"; "Mary"; "lastname"; "Smith"; "schoolName"; "Math school")

$status:=$remoteDS.Schools.registerNewStudent($student) // OK
$id:=$remoteDS.Schools.computeIDNumber() // Error "Unknown member method"

```

## Funciones locales

By default in client/server architecture, ORDA data model functions are executed on the server. It usually provides the best performance since only the function request and the result are sent over the network.

However, it could happen that a function is fully executable on the client side (e.g., when it processes data that's already in the local cache). In this case, you can save requests to the server and thus, enhance the application performance by inserting the `local` keyword. La sintaxis formal es:

```
// declare a function to execute locally in client/server
local Function <functionName>
```

With this keyword, the function will always be executed on the client side.

The `local` keyword can only be used with data model class functions. If used with a [regular user class](#) function, it is ignored and an error is returned by the compiler.

Note that the function will work even if it eventually requires to access the server (for example if the ORDA cache is expired). However, it is highly recommended to make sure that the local function does not access data on the server, otherwise the local execution could not bring any performance benefit. A local function that generates many requests to the server is less efficient than a function executed on the server that would only return the resulting values. For example, consider the following function on the Schools entity class:

```

// Get the youngest students
// Inappropriate use of local keyword
local Function getYoungest
    var $0 : Object
    $0:=This.students.query("birthDate >= :1"; !2000-01-01!).orderBy("birthDate desc").slice(0; 5)

```

- without the `local` keyword, the result is given using a single request
- with the `local` keyword, 4 requests are necessary: one to get the Schools entity students, one for the `query()`, one for the `orderBy()`, and one for the `slice()`. In this example, using the `local` keyword is inappropriate.

## Ejemplos

### Cálculo de la edad

Given an entity with a `birthDate` attribute, we want to define an `age()` function that would be called in a list box. This function can be executed on the client, which avoids triggering a request to the server for each line of the list box.

En la clase `StudentsEntity`:

```

Class extends Entity

local Function age() -> $age: Variant

If (This.birthDate#!00-00-00!)
    $age:=Year of(Current date)-Year of(This.birthDate)
Else
    $age:=Null
End if
    $age:=Year of(Current date)-Year of(This.birthDate)
Else
    $age:=Null
End if

```

### Verificación de los atributos

We want to check the consistency of the attributes of an entity loaded on the client and updated by the user before requesting the server to save them.

On the `StudentsEntity` class, the local `checkData()` function checks the Student's age:

```

Class extends Entity

local Function checkData() -> $status : Object

$status:=New object("success"; True)
Case of
    : (This.age()=Null)
        $status.success:=False
        $status.statusText:="The birthdate is missing"

    :((This.age() <15) | (This.age()>30) )
        $status.success:=False
        $status.statusText:="The student must be between 15 and 30 - This one is "+String(This.age())
End case

```

Código de llamada:

```

var $status : Object

//Form.student is loaded with all its attributes and updated on a Form
$status:=Form.student.checkData()
If ($status.success)
    $status:=Form.student.save() // call the server
End if

```

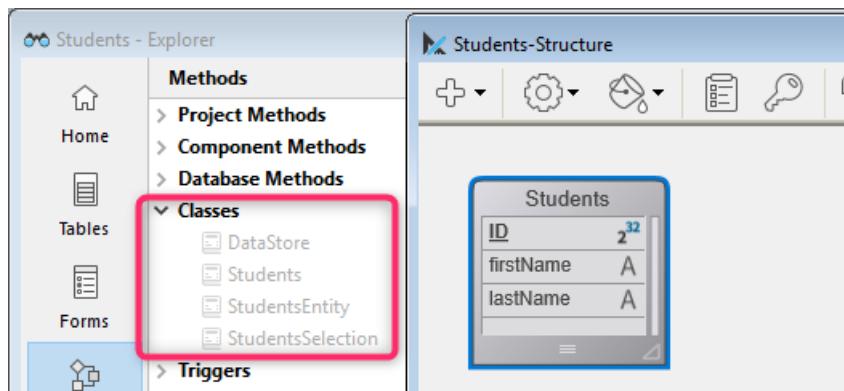
## Soporte en 4D IDE

### Archivos de clase (class files)

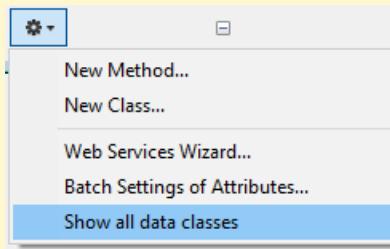
An ORDA data model user class is defined by adding, at the [same location as regular class files](#) (i.e. in the `/Sources/Classes` folder of the project folder), a `.4dm` file with the name of the class. For example, an entity class for the `Utilities` dataclass will be defined through a `UtilitiesEntity.4dm` file.

### Crear clases

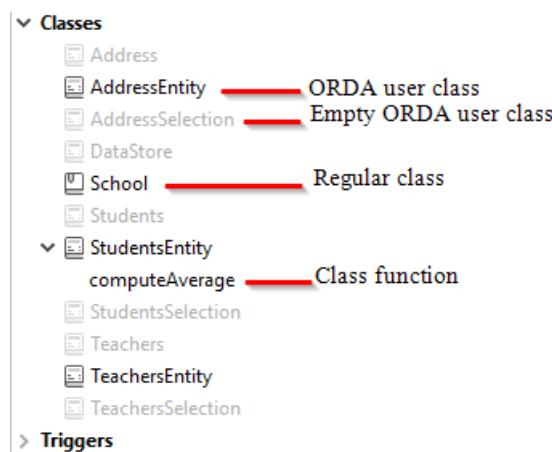
4D automatically pre-creates empty classes in memory for each available data model object.



By default, empty ORDA classes are not displayed in the Explorer. To show them you need to select [Show all data classes](#) from the Explorer's options menu:



ORDA user classes have a different icon from regular classes. Las clases vacías se atenúan:



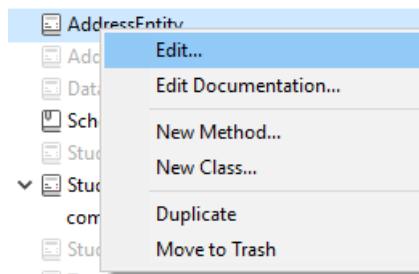
To create an ORDA class file, you just need to double-click on the corresponding predefined class in the Explorer. 4D creates the class file and add the `extends` code. For example, for an Entity class:

**Class extends Entity**

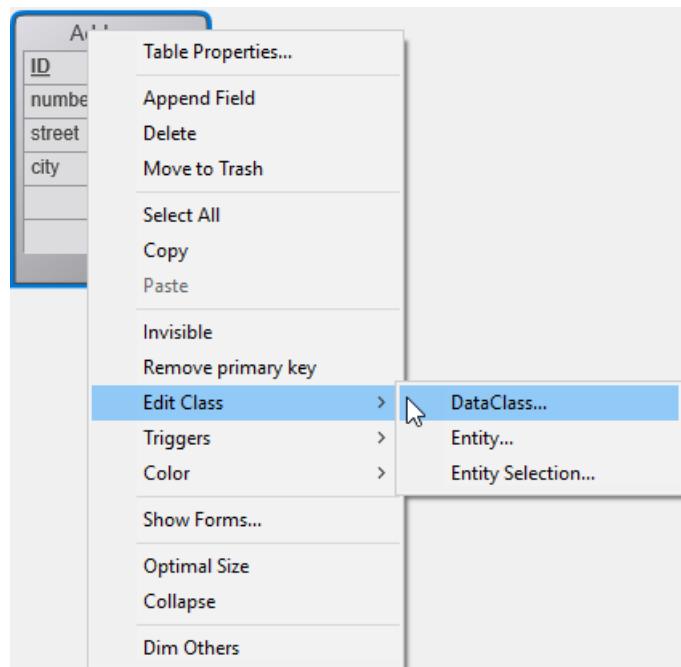
Once a class is defined, its name is no longer dimmed in the Explorer.

## Editar las clases

To open a defined ORDA class in the 4D method editor, select or double-click on an ORDA class name and use `Edit...` from the contextual menu/options menu of the Explorer window:



For ORDA classes based upon the local datastore (`ds`), you can directly access the class code from the 4D Structure window:



## Editor de método

In the 4D method editor, variables typed as an ORDA class automatically benefit from autocompletion features. Ejemplo con una variable de clase Entity:

```
var $student : cs.StudentsEntity
$student:=ds.Students.all().first()
$student.|
```

The screenshot shows the 4D Method Editor with the following code snippet:

```
var $student : cs.StudentsEntity
$student:=ds.Students.all().first()
$student.|
```

An autocompletion dropdown is open at the cursor position. The suggestions include: ID, name, clone, computeAverage (which is highlighted with a red box), diff, drop, first, and fromObject.



# Trabajar con los datos

In ORDA, you access data through [entities](#) and [entity selections](#). These objects allow you to create, update, query, or sort the data of the datastore.

## Crear una entidad

There are two ways to create a new entity in a dataclass:

- Since entities are references to database records, you can create entities by creating records using the "classic" 4D language and then reference them with ORDA methods such as `entity.next()` or `entitySelection.first()`.
- You can also create an entity using the `dataClass.new()` method.

Keep in mind that the entity is only created in memory. If you want to add it to the datastore, you must call the `entity.save()` method.

Entity attributes are directly available as properties of the entity object. For more information, please refer to [Using entity attributes](#).

For example, we want to create a new entity in the "Employee" dataclass in the current datastore with "John" and "Dupont" assigned to the `firstname` and `name` attributes:

```
var $myEntity : cs.EmployeeEntity
$myEntity:=ds.Employee.new() //Create a new object of the entity type
$myEntity.name:="Dupont" //assign 'Dupont' to the 'name' attribute
$myEntity.firstname:="John" //assign 'John' to the 'firstname' attribute
$myEntity.save() //save the entity
```

An entity is defined only in the process where it was created. You cannot, for example, store a reference to an entity in an interprocess variable and use it in another process.

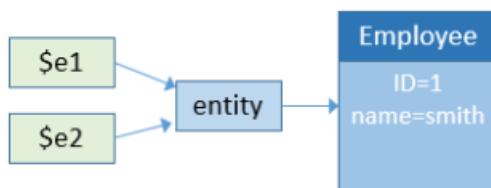
## Entidades y referencias

Una entidad contiene una referencia a un registro 4D. Different entities can reference the same 4D record. Also, since an entity can be stored in a 4D object variable, different variables can contain a reference to the same entity.

Si ejecuta el siguiente código:

```
var $e1; $e2 : cs.EmployeeEntity
$e1:=ds.Employee.get(1) //access the employee with ID 1
$e2:=$e1
$e1.name:="Hammer"
//both variables $e1 and $e2 share the reference to the same entity
//$/e2.name contains "Hammer"
```

Esto es ilustrado por el siguiente gráfico:



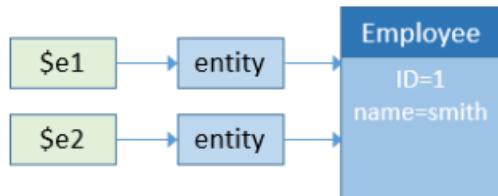
Ahora, si se ejecuta:

```

var $e1; $e2 : cs.EmployeeEntity
$e1:=ds.Employee.get(1)
$e2:=ds.Employee.get(1)
$e1.name:="Hammer"
//variable $e1 contains a reference to an entity
//variable $e2 contains another reference to another entity
//$e2.name contains "smith"

```

Esto es ilustrado por el siguiente gráfico:



Note however that entities refer to the same record. In all cases, if you call the `entity.save( )` method, the record will be updated (except in case of conflict, see [Entity locking](#)).

In fact, `$e1` and `$e2` are not the entity itself, but a reference to the entity. It means that you can pass them directly to any function or method, and it will act like a pointer, and faster than a 4D pointer. Por ejemplo:

```

For each($entity;$selection)
    do_Capitalize($entity)
End for each

```

Y el método es:

```

$entity:=$1
$name:=$entity.lastname
If(Not($name=NULL))
    $name:=Uppercase(Substring($name;1;1))+Lowercase(Substring($name;2))
End if
$entity.lastname:=$name

```

You can handle entities like any other object in 4D and pass their references directly as [parameters](#).

With the entities, there is no concept of "current record" as in the classic 4D language. You can use as many entities as you need, at the same time. There is also no automatic lock on an entity (see [Entity locking](#)). When an entity is loaded, it uses the [lazy loading](#) mechanism, which means that only the needed information is loaded. Nevertheless, in client/server, the entity can be automatically loaded directly if necessary.

## Uso de los atributos de entidades

Entity attributes store data and map corresponding fields in the corresponding table. Entity attributes of the storage kind can be set or get as simple properties of the entity object, while entity of the relatedEntity or relatedEntities kind will return an entity or an entity selection.

For more information on the attribute kind, please refer to the [Storage and Relation attributes](#) paragraph.

Por ejemplo, para definir un atributo de almacenamiento:

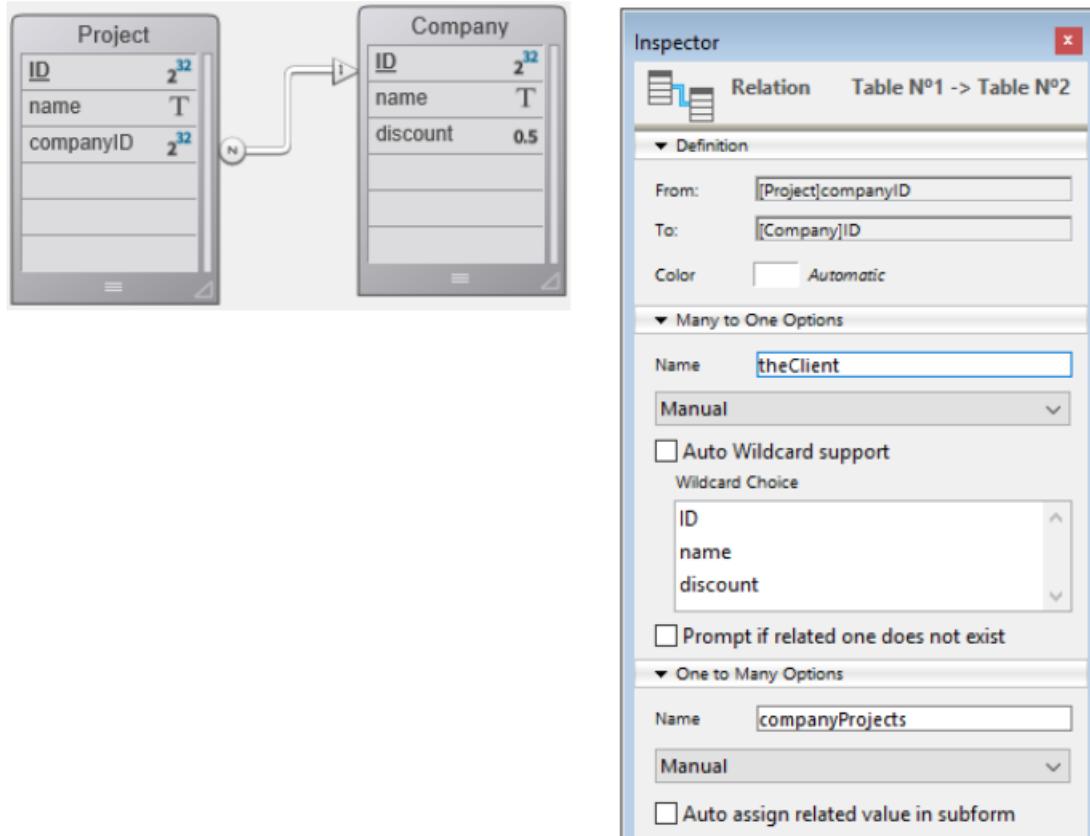
```

$entity:=ds.Employee.get(1) //get employee attribute with ID 1
$name:=$entity.lastname //get the employee name, e.g. "Smith"
$entity.lastname:="Jones" //set the employee name
$entity.save() //save the modifications

```

Database Blob fields ([scalar blobs](#)) are automatically converted to and from blob object attributes ( [4D.Blob](#) ) when handled through ORDA. When saving a blob object attribute, keep in mind that, unlike blob object size which is only limited by the available memory, Blob field size is limited to 2GB.

Accessing a related attribute depends on the attribute kind. Por ejemplo, con la siguiente estructura:



Puede acceder a los datos a través del objeto(s) relacionado(s):

```

$entity:=ds.Project.all().first().theClient //get the Company entity associated to the project
$EntitySel:=ds.Company.all().first().companyProjects //get the selection of projects for the company

```

Note that both `theClient` and `companyProjects` in the above example are primary relation attributes and represent a direct relationship between the two dataclasses. However, relation attributes can also be built upon paths through relationships at several levels, including circular references. Por ejemplo, consideremos la siguiente estructura:

Each employee can be a manager and can have a manager. To get the manager of the manager of an employee, you can simply write:

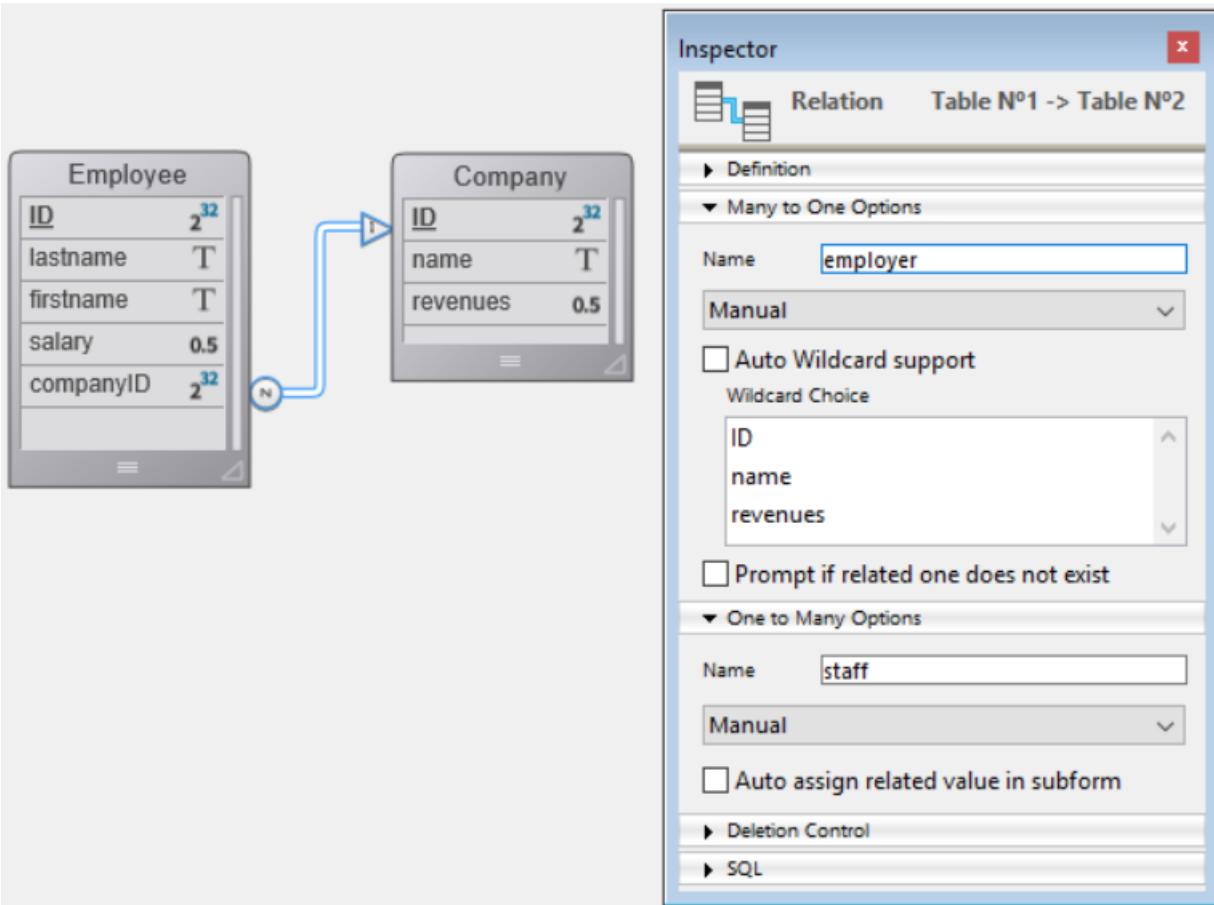
```
$myEmp:=ds.Employee.get(50)
$manLev2:=$myEmp.manager.manager.lastname
```

## Asignar los valores a los atributos de relación

In the ORDA architecture, relation attributes directly contain data related to entities:

- An N->1 type relation attribute (relatedEntity kind) contains an entity
- A 1->N type relation attribute (relatedEntities kind) contains an entity selection

Let's look at the following (simplified) structure:



In this example, an entity in the "Employee" dataclass contains an object of type Entity in the "employer" attribute (or a null value). An entity in the "Company" dataclass contains an object of type EntitySelection in the "staff" attribute (or a null value).

In ORDA, the Automatic or Manual property of relations has no effect.

To assign a value directly to the "employer" attribute, you must pass an existing entity from the "Company" dataclass. Por ejemplo:

```
$emp:=ds.Employee.new() // create an employee
$emp.lastname:="Smith" // assign a value to an attribute
$emp.employer:=ds.Company.query("name =:1";"4D")[0] //assign a company entity
$emp.save()
```

4D provides an additional facility for entering a relation attribute for an N entity related to a "1" entity: you pass the primary key of the "1" entity directly when assigning a value to the relation attribute. For this to work, you pass data of type Number or Text (the primary key value) to the relation attribute. 4D then automatically takes care of searching for the corresponding entity in the dataclass. Por ejemplo:

```
$emp:=ds.Employee.new()
$emp.lastname:="Wesson"
$emp.employer:=2 // assign a primary key to the relation attribute
//4D looks for the company whose primary key (in this case, its ID) is 2
//and assigns it to the employee
$emp.save()
```

This is particularly useful when you are importing large amounts of data from a relational database. This type of import usually contains an "ID" column, which references a primary key that you can then assign directly to a relation attribute.

This also means that you can assign primary keys in the N entities without corresponding entities having already been created in the 1 datastore class. If you assign a primary key that does not exist in the related datastore class, it is

nevertheless stored and assigned by 4D as soon as this "1" entity is created.

You can assign or modify the value of a "1" related entity attribute from the "N" dataclass directly through the related attribute. For example, if you want to modify the name attribute of a related Company entity of an Employee entity, you can write:

```
$emp:=ds.Employee.get(2) // load the Employee entity with primary key 2  
$emp.employer.name:="4D, Inc." //modify the name attribute of the related Company  
$emp.employer.save() //save the related attribute  
//the related entity is updated
```

## Crear una entidad seleccion

You can create an object of type [entity selection](#) as follows:

- Querying the entities [in a dataclass](#) or in an [existing entity selection](#);
- Using the [.all\(\)](#) dataclass function to select all the entities in a dataclass;
- Using the [Create entity selection](#) command or the [.newSelection\(\)](#) dataclass function to create a blank entity selection;
- Using the [.copy\(\)](#) function to duplicate an existing entity selection;
- Using one of the various functions from the [Entity selection class](#) that returns a new entity selection, such as [.or\(\)](#) ;
- Using a relation attribute of type "related entities" (see below).

You can simultaneously create and use as many different entity selections as you want for a dataclass. Keep in mind that an entity selection only contains references to entities. Different entity selections can contain references to the same entities.

## Entity selections compatibles o modificables

An entity selection can be shareable (readable by multiple processes, but not alterable after creation) or alterable (supports the [.add\(\)](#) function, but only usable by the current process).

### Propiedades

A shareable entity selection has the following characteristics:

- it can be stored in a shared object or shared collection, and can be passed as parameter between several processes or workers;
- it can be stored in several shared objects or collections, or in a shared object or collection which already belongs to a group (it does not have a *locking identifier*);
- no permite la adición de nuevas entidades. Trying to add an entity to a shareable entity selection will trigger an error (1637 - This entity selection cannot be altered). To add an entity to a shareable entity selection, you must first transform it into a non-shareable entity selection using the [.copy\(\)](#) function, before calling [.add\(\)](#) .

Most entity selection functions (such as [.slice\(\)](#) , [.and\(\)](#) ...) support shareable entity selections since they do not need to alter the original entity selection (they return a new one).

An alterable entity selection has the following characteristics:

- it cannot be shared between processes, nor be stored in a shared object or collection. Trying to store a non-shareable entity selection in a shared object or collection will trigger an error (-10721 - Not supported value type in a shared object or shared collection);
- it accepts the addition of new entities, i.e. it supports the [.add\(\)](#) function.

### ¿Cómo se definen?

The shareable or alterable nature of an entity selection is defined when the entity selection is created (it cannot be

modified afterwards). You can know the nature of an entity selection using the `.isAlterable()` function or the `OB Is shared` command.

A new entity selection is shareable in the following cases:

- the new entity selection results from an ORDA class function applied to a dataClass: `dataClass.all()`, `dataClass.fromCollection()`, `dataClass.query()`,
- the new entity selection results from one of the various ORDA class functions applied to an existing entity selection (`.query()`, `.slice()`, etc.) .
- the new entity selection is explicitly copied as shareable with `entitySelection.copy()` or `OB Copy` (i.e. with the `ck shared` option).

Ejemplo:

```
$myComp:=ds.Company.get(2) //$/myComp does not belong to an entity selection  
$employees:=$myComp.employees //$/employees is shareable
```

A new entity selection is alterable in the following cases:

- the new entity selection created blank using the `dataClass.newSelection()` function or `Create entity selection` command,
- the new entity selection is explicitly copied as alterable with `entitySelection.copy()` or `OB Copy` (i.e. without the `ck shared` option).

Ejemplo:

```
$toModify:=ds.Company.all().copy() //$/toModify is alterable
```

A new entity selection inherits from the original entity selection nature in the following cases:

- the new entity selection is based upon a relation `entity.attributeName` (e.g. `Employee.employer`),
- la nueva selección de entidades se basa en una relación:
  - `entity.attributeName` (e.g. "company.employees") when `attributeName` is a one-to-many related attribute and the entity belongs to an entity selection (same nature as `.getSelection()` entity selection),
  - `entitySelection.attributeName` (e.g. "employees.employer") when `attributeName` is a related attribute (same nature as the entity selection),
  - `.extract()` when the resulting collection contains entity selections (same nature as the entity selection).

Ejemplos:

```
$highSal:=ds.Employee.query("salary >= :1"; 1000000)  
    //$/highSal is shareable because of the query on dataClass  
$comp:=$highSal.employer //$/comp is shareable because $highSal is shareable  
  
$lowSal:=ds.Employee.query("salary <= :1"; 10000).copy()  
    //$/lowSal is alterable because of the copy()  
$comp2:=$lowSal.employer //$/comp2 is alterable because $lowSal is alterable
```

## Sharing an entity selection between processes (example)

You work with two entity selections that you want to pass to a worker process so that it can send mails to appropriate persons:

```

var $paid; $unpaid : cs.InvoicesSelection
//We get entity selections for paid and unpaid invoices
$paid:=ds.Invoices.query("status=:1"; "Paid")
$unpaid:=ds.Invoices.query("status=:1"; "Unpaid")

//We pass entity selection references as parameters to the worker
CALL WORKER("mailing"; "sendMails"; $paid; $unpaid)

```

El método `sendMails` :

```

#DECLARE ($paid : cs.InvoicesSelection; $unpaid : cs.InvoicesSelection)
var $invoice : cs.InvoicesEntity

var $server; $transporter; $email; $status : Object

//Prepare emails
$server:=New object()
$server.host:="exchange.company.com"
$server.user:="myName@company.com"
$server.password:="my!password"
$transporter:=SMTP New transporter($server)
$email:=New object()
$email.from:="myName@company.com"

//Loops on entity selections
For each($invoice;$paid)
    $email.to:=$invoice.customer.address // email address of the customer
    $email.subject:="Payment OK for invoice # "+String($invoice.number)

    $status:=$transporter.send($email)
End for each

For each($invoice;$unpaid)
    $email.to:=$invoice.customer.address // email address of the customer
    $email.subject:="Please pay invoice # "+String($invoice.number)
    $status:=$transporter.send($email)
End for each

```

## Selecciones de entidades y atributos de almacenamiento

All storage attributes (text, number, boolean, date) are available as properties of entity selections as well as entities. When used in conjunction with an entity selection, a scalar attribute returns a collection of scalar values. Por ejemplo:

```

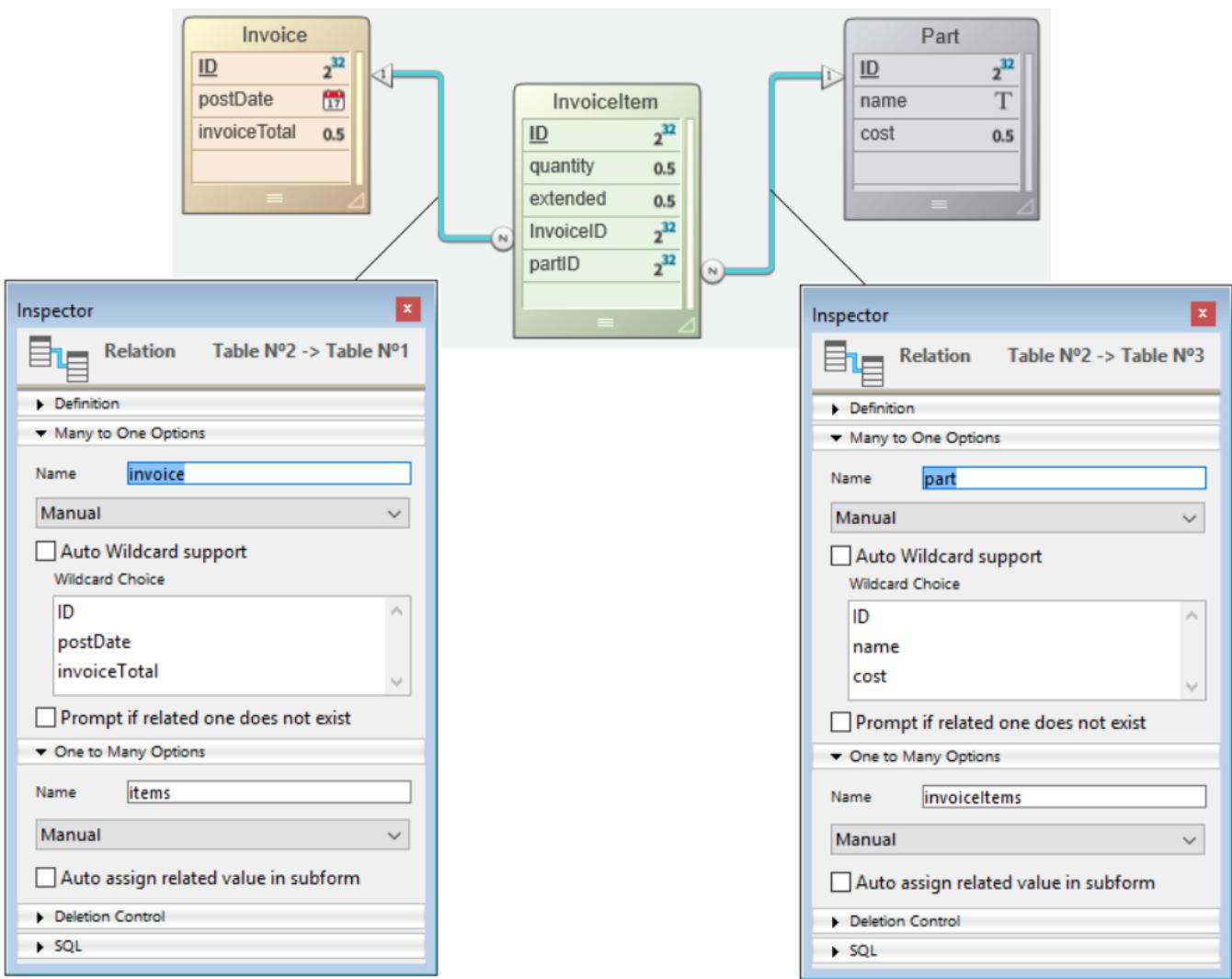
$locals:=ds.Person.query("city = :1";"San Jose") //entity selection of people
$localEmails:=$locals.emailAddress //collection of email addresses (strings)

```

Este código devuelve en `$localEmails` una colección de direcciones de correo electrónico como cadenas.

## Selecciones de entidades y atributos de relación

In addition to the variety of ways you can query, you can also use relation attributes as properties of entity selections to return new entity selections. Por ejemplo, consideremos la siguiente estructura:



```
$myParts:=ds.Part.query("ID < 100") //Return parts with ID less than 100
$myInvoices:=$myParts.invoiceItems.invoice
//All invoices with at least one line item related to a part in $myParts
```

The last line will return in \$myInvoices an entity selection of all invoices that have at least one invoice item related to a part in the entity selection myParts. When a relation attribute is used as a property of an entity selection, the result is always another entity selection, even if only one entity is returned. When a relation attribute is used as a property of an entity selection and no entities are returned, the result is an empty entity selection, not null.

## Bloqueo de una entidad

You often need to manage possible conflicts that might arise when several users or processes load and attempt to modify the same entities at the same time. Record locking is a methodology used in relational databases to avoid inconsistent updates to data. The concept is to either lock a record upon read so that no other process can update it, or alternatively, to check when saving a record to verify that some other process hasn't modified it since it was read. The former is referred to as pessimistic record locking and it ensures that a modified record can be written at the expense of locking records to other users. The latter is referred to as optimistic record locking and it trades the guarantee of write privileges to the record for the flexibility of deciding write privileges only if the record needs to be updated. In pessimistic record locking, the record is locked even if there is no need to update it. In optimistic record locking, the validity of a record's modification is decided at update time.

ORDA le ofrece dos modos de bloqueo de entidad:

- an automatic "optimistic" mode, suitable for most applications,
- a "pessimistic" mode allowing you to lock entities prior to their access.

### Bloqueo automático optimista

This automatic mechanism is based on the concept of "optimistic locking" which is particularly suited to the issues of web applications. This concept is characterized by the following operating principles:

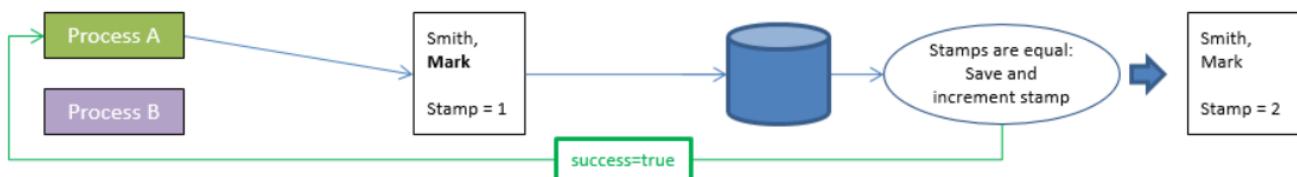
- All entities can always be loaded in read-write; there is no *a priori* "locking" of entities.
- Each entity has an internal locking stamp that is incremented each time it is saved.
- When a user or process tries to save an entity using the `entity.save()` method, 4D compares the stamp value of the entity to be saved with that of the entity found in the data (in the case of a modification):
  - When the values match, the entity is saved and the internal stamp value is incremented.
  - When the values do not match, it means that another user has modified this entity in the meantime. The save is not performed and an error is returned.

The following diagram illustrates optimistic locking:

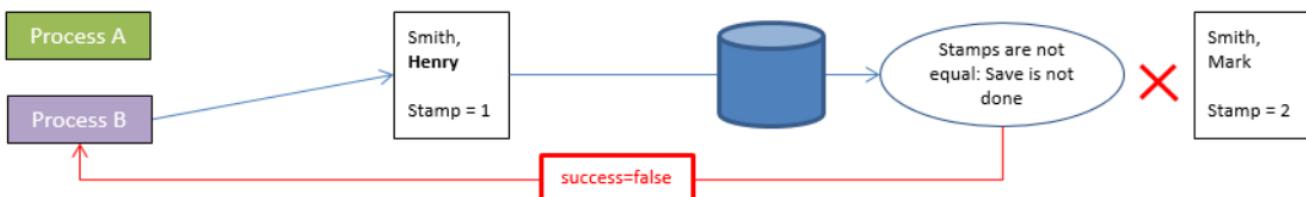
1. Dos procesos cargan la misma entidad.



2. The first process modifies the entity and validates the change. Se llama al método `entity.save()`. The 4D engine automatically compares the internal stamp value of the modified entity with that of the entity stored in the data. Since they match, the entity is saved and its stamp value is incremented.



3. The second process also modifies the loaded entity and validates its changes. Se llama al método `entity.save()`. Since the stamp value of the modified entity does not match the one of the entity stored in the data, the save is not performed and an error is returned.



This can also be illustrated by the following code:

```

$person1:=ds.Person.get(1) //Reference to entity
$person2:=ds.Person.get(1) //Other reference to same entity
$person1.name:="Bill"
$result:=$person1.save() //$/result.success=true, change saved
$person2.name:="William"
$result:=$person2.save() //$/result.success=false, change not saved
    
```

En este ejemplo, asignamos a \$person1 una referencia a la entidad person con una llave de 1. Then, we assign another reference of the same entity to variable \$person2. Using \$person1, we change the first name of the person and save the entity. When we attempt to do the same thing with \$person2, 4D checks to make sure the entity on disk is the same as when the reference in \$person1 was first assigned. Since it isn't the same, it returns false in the success property and doesn't save the second modification.

When this situation occurs, you can, for example, reload the entity from the disk using the `entity.reload()` method so that you can try to make the modification again. The `entity.save()` method also proposes an "automerge" option to save the entity in case processes modified attributes that were not the same.

Record stamps are not used in transactions because only a single copy of a record exists in this context.

Whatever the number of entities that reference a record, the same copy is modified thus `entity.save()` operations will never generate stamp errors.

## Bloqueo pesimista

You can lock and unlock entities on demand when accessing data. When an entity is getting locked by a process, it is loaded in read/write in this process but it is locked for all other processes. The entity can only be loaded in read-only mode in these processes; its values cannot be edited or saved.

This feature is based upon two functions of the `Entity` class:

- `entity.lock()`
- `entity.unlock()`

For more information, please refer to the descriptions for these functions.

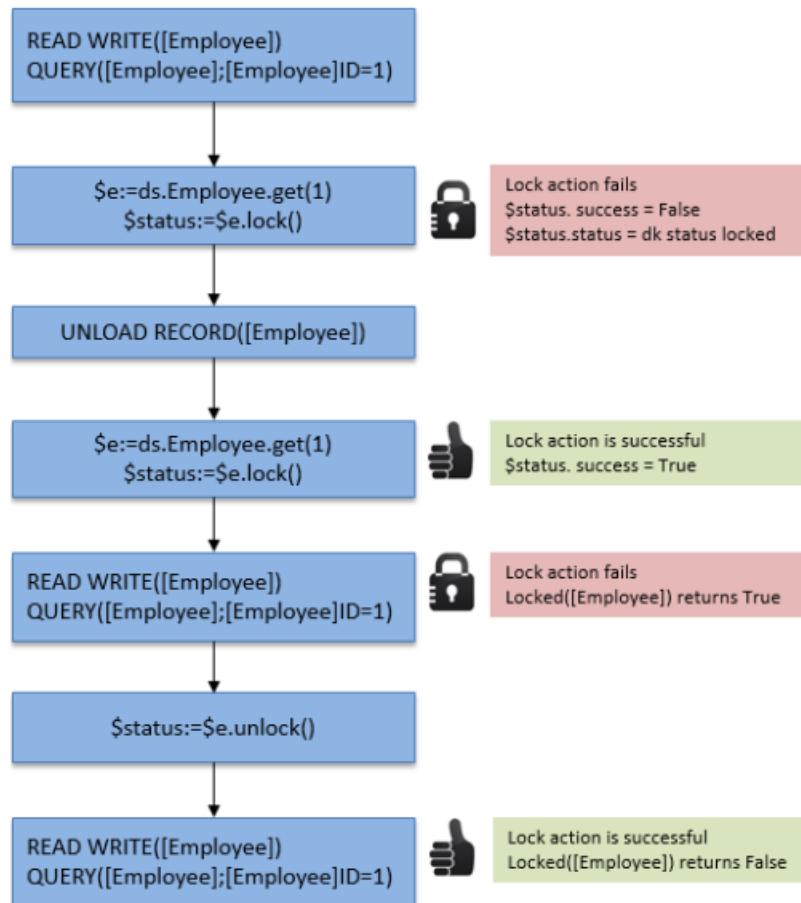
Pessimistic locks can also be handled through the [REST API](#).

## Concurrent use of 4D classic locks and ORDA pessimistic locks

Using both classic and ORDA commands to lock records is based upon the following principles:

- A lock set with a classic 4D command on a record prevents ORDA to lock the entity matching the record.
- A lock set with ORDA on an entity prevents classic 4D commands to lock the record matching the entity.

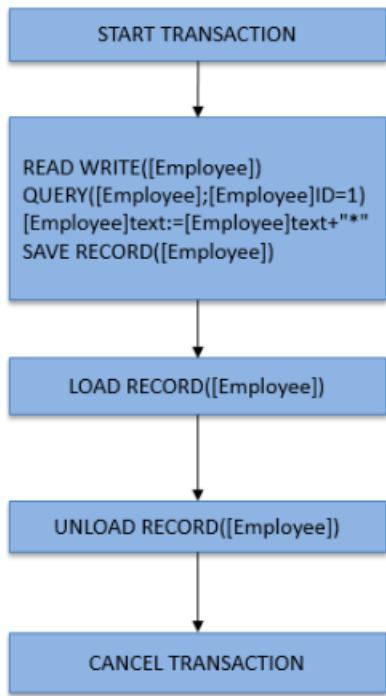
These principles are shown in the following diagram:



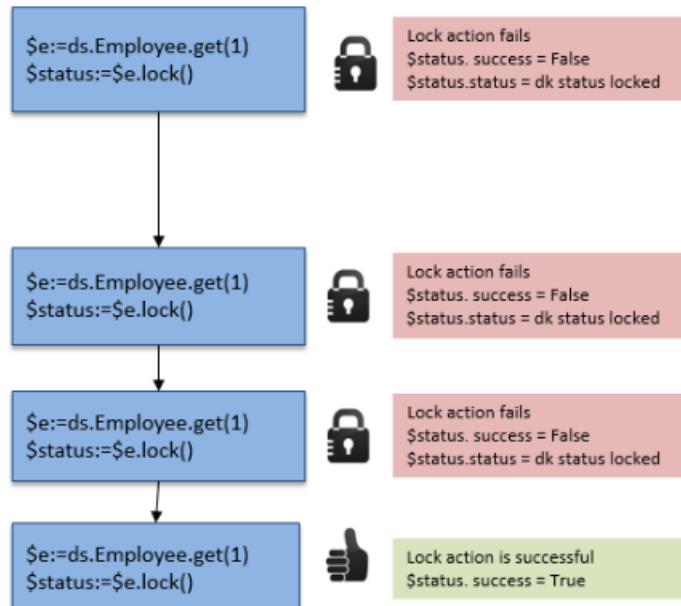
Transaction locks also apply to both classic and ORDA commands. In a multiprocess or a multi-user application, a lock set within a transaction on a record by a classic command will result in preventing any other processes to lock entities related to this record (or conversely), until the transaction is validated or canceled.

- Ejemplo con un bloqueo definido por un comando clásico:

## Process P1

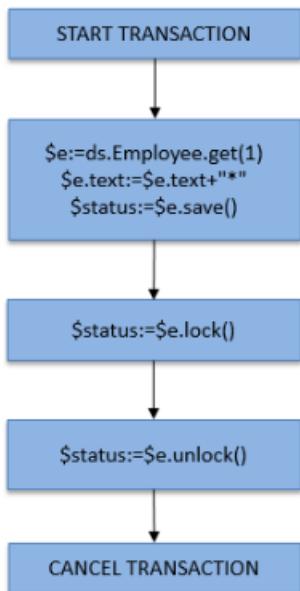


## Process P2

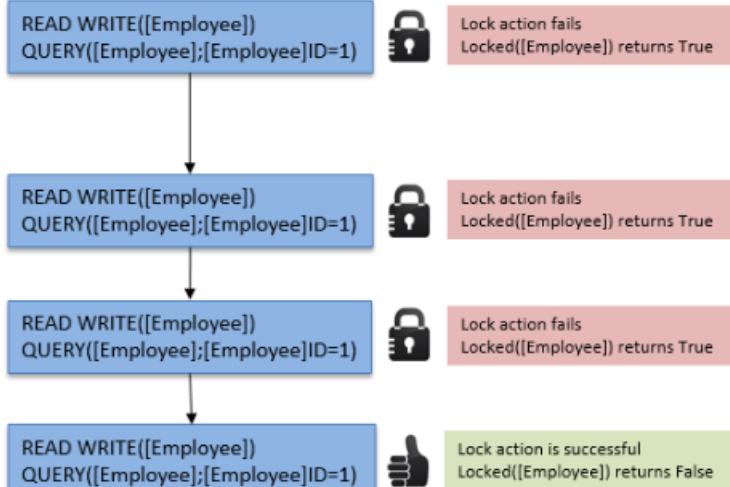


- Example with a lock set by an ORDA function:

## Process P1



## Process P2



# Utilizar un almacén de datos remoto

A [datastore](#) exposed on a 4D application can be accessed simultaneously through different clients:

- 4D remote applications using ORDA to access the main datastore with the `ds` command. Note that the 4D remote application can still access the database in classic mode. These accesses are handled by the 4D application server.
- Other 4D applications (4D remote, 4D Server) opening a session on the remote datastore through the `open datastore` command. These accesses are handled by the HTTP REST server.
- [4D for iOS or 4D for Android](#) queries for updating mobile applications. These accesses are handled by the HTTP server.

## Apertura de las sesiones

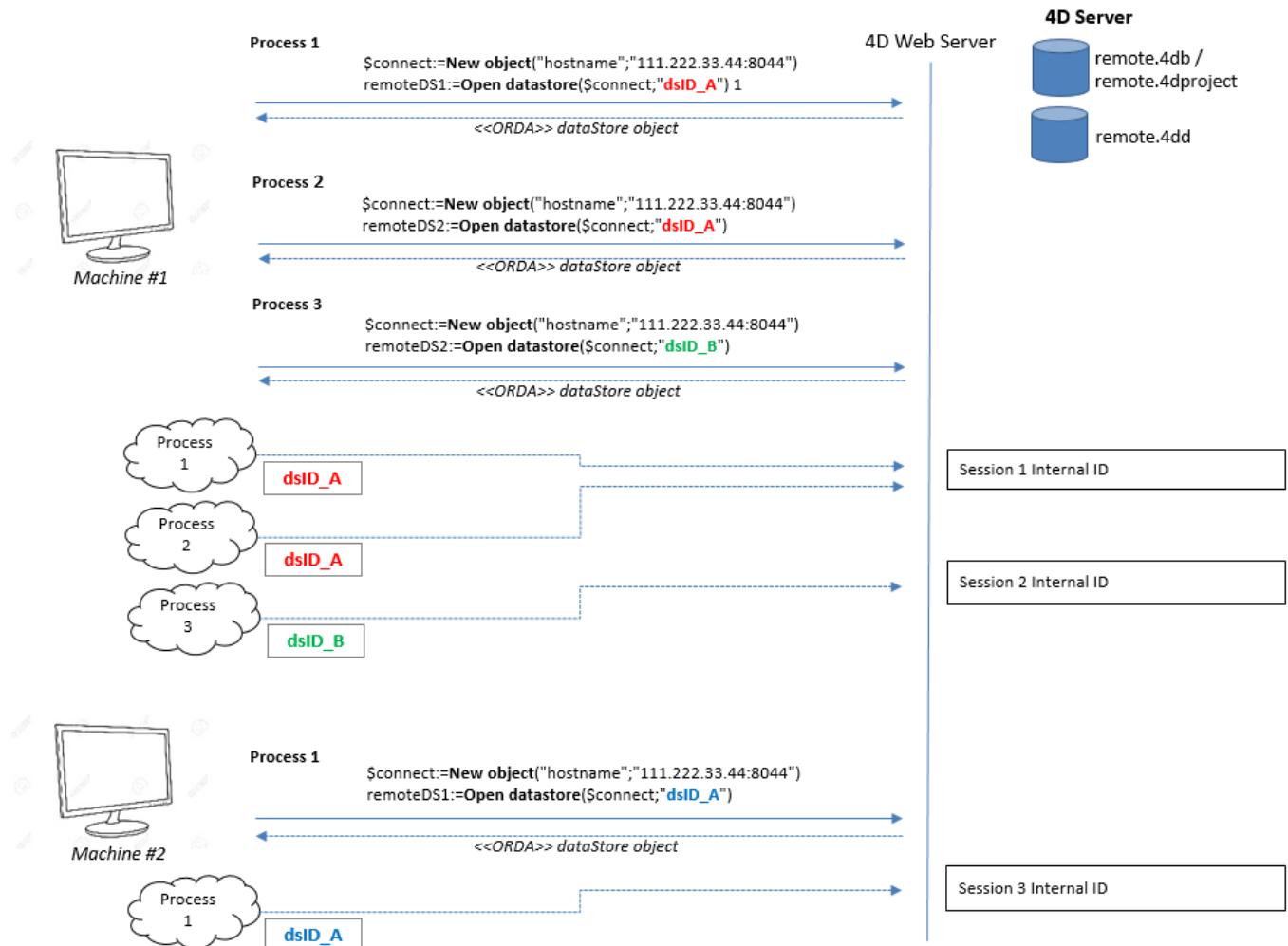
When you work with a remote datastore referenced through calls to the `Open datastore` command, the connection between the requesting processes and the remote datastore is handled via sessions.

A session is created on the remote datastore to handle the connection. This session is identified using a internal session ID which is associated to the `localID` on the 4D application side. This session automatically manages access to data, entity selections, or entities.

The `localID` is local to the machine that connects to the remote datastore, which means:

- If other processes of the same application need to access the same remote datastore, they can use the same `localID` and thus, share the same session.
- If another process of the same application opens the same remote datastore but with another `localID`, it will create a new session on the remote datastore.
- If another machine connects to the same remote datastore with the same `localID`, it will create another session with another cookie.

These principles are illustrated in the following graphics:



For sessions opened by REST requests, please refer to [Users and sessions](#).

## Visualización de las sesiones

Processes that manage sessions for datastore access are shown in the 4D Server administration window:

- nombre: "REST Handler: <process name>"
- type: HTTP Server Worker type
- session: session name is the user name passed to the `Open datastore` command.

In the following example, two processes are running for the same session:

EmpComp - 4D Server Administration							
Monitor	Users (1)	Processes (23)	Maintenance	Application Server	SQL Server	HTTP Server	Real Time Monitor
<input checked="" type="checkbox"/> Display processes by groups				Users processes (0)	4D Processes (16)	Spare processes (7)	
Process name	Session / Info	Type	Num	State	CPU Time	Activity	
Client Manager	-	Application server	3	Waiting for event	00:00:01	0 %	
DB4D CRON	-	DB4D Server	0	Running	00:00:01	0 %	
DB4D Flush	-	DB4D Server	0	Running	00:00:01	0 %	
DB4D Index builder	-	DB4D Server	0	Running	00:00:01	0 %	
DB4D Server	-	DB4D Server	0	Running	00:00:01	0 %	
DB4D Sockets	-	DB4D Server	0	Running	00:00:01	0 %	
Garbage Handler	-	DB4D Server	0	Running	00:00:01	0 %	
HTTP Listener	-	Web Server	0	Running	00:00:01	0 %	
Internal Timer Process	-	Application server	2	Executing	00:00:01	0 %	
Logger	-	Logger process	0	Running	00:00:01	0 %	
Task managers	-	SQL Server	0	Running	00:00:01	0 %	
TCP connection listener	-	TCP Connection listener	0	Running	00:00:01	0 %	
TCP connection listener	-	SQL Server	0	Running	00:00:01	0 %	
User Interface	-	Application server	1	Waiting for event	00:00:02	2 %	
REST Handler: process1	marie-sophie	HTTP Server Worker	0	Running	00:00:08	90 %	
REST Handler: process2	marie-sophie	HTTP Server Worker	0	Running	00:00:08	89 %	

## Bloqueo y transacciones

ORDA features related to entity locking and transaction are managed at process level in remote datastores, just like in ORDA client/server mode:

- If a process locks an entity from a remote datastore, the entity is locked for all other processes, even when these processes share the same session (see [Entity locking](#)). If several entities pointing to a same record have been locked in a process, they must be all unlocked in the process to remove the lock. If a lock has been put on an entity, the lock is removed when there is no more reference to this entity in memory.
- Transactions can be started, validated or cancelled separately on each remote datastore using the `dataStore.startTransaction()`, `dataStore.cancelTransaction()`, and `dataStore.validateTransaction()` functions. No afectan a otros almacenes de datos.
- Classic 4D language commands (`START TRANSACTION`, `VALIDATE TRANSACTION`, `CANCEL TRANSACTION`) only apply to the main datastore (returned by `ds`). If an entity from a remote datastore is hold by a transaction in a process, other processes cannot update it, even if these processes share the same session.
- Locks on entities are removed and transactions are rolled back:
  - when the process is killed.
  - cuando la sesión se cierra en el servidor
  - when the session is killed from the server administration window.

## Cierre de las sesiones

A session is automatically closed by 4D when there has been no activity during its timeout period. The default timeout is 60 mn, but this value can be modified using the `connectionInfo` parameter of the `Open` datastore command.

If a request is sent to the remote datastore after the session has been closed, it is automatically re-created if possible (license available, server not stopped...). However, keep in mind that the context of the session regarding locks and transactions is lost (see above).

## Optimización cliente/servidor

4D provides an automatic optimization for ORDA requests that use entity selections or load entities in client/server configurations (datastore accessed remotely through `ds` or via `Open` datastore). This optimization speeds up the execution of your 4D application by reducing drastically the volume of information transmitted over the network.

The following optimization mechanisms are implemented:

- When a client requests an entity selection from the server, 4D automatically "learns" which attributes of the entity selection are actually used on the client side during the code execution, and builds a corresponding "optimization

context". This context is attached to the entity selection and stores the used attributes. It will be dynamically updated if other attributes are used afterwards.

- Subsequent requests sent to the server on the same entity selection automatically reuse the optimization context and only get necessary attributes from the server, which accelerates the processing. For example in an entity selection-based list box, the learning phase takes place during the display of the first rows, next rows display is very optimized.
- An existing optimization context can be passed as a property to another entity selection of the same dataclass, thus bypassing the learning phase and accelerating the application (see [Using the context property](#) below).

The following methods automatically associate the optimization context of the source entity selection to the returned entity selection:

- `entitySelection.and()`
- `entitySelection.minus()`
- `entitySelection.or()`
- `entitySelection.orderBy()`
- `entitySelection.slice()`
- `entitySelection.drop()`

#### Ejemplo

Dado el siguiente código:

```
$sel:=$ds.Employee.query("firstname = ab@")
For each($e;$sel)
    $s:=$e.firstname+" "+$e.lastname+" works for "+$e.employer.name // $e.employer refers to Company tab
End for each
```

Thanks to the optimization, this request will only get data from used attributes (firstname, lastname, employer, employer.name) in `$sel` after a learning phase.

## Uso de la propiedad context

You can increase the benefits of the optimization by using the `context` property. This property references an optimization context "learned" for an entity selection. It can be passed as parameter to ORDA methods that return new entity selections, so that entity selections directly request used attributes to the server and bypass the learning phase.

A same optimization context property can be passed to unlimited number of entity selections on the same dataclass. All ORDA methods that handle entity selections support the context property (for example `dataClass.query( )` or `dataClass.all( )` method). Keep in mind, however, that a context is automatically updated when new attributes are used in other parts of the code. Reusing the same context in different codes could result in overloading the context and then, reduce its efficiency.

A similar mechanism is implemented for entities that are loaded, so that only used attributes are requested (see the `dataClass.get( )` method).

Example with `dataClass.query( )` method:

```

var $sel1; $sel2; $sel3; $sel4; $querysettings; $querysettings2 : Object
var $data : Collection
$querysettings:=New object("context";"shortList")
$querysettings2:=New object("context";"longList")

$sel1:=ds.Employee.query("lastname = S@";$querysettings)
$data:=extractData($sel1) // In extractData method an optimization is triggered and associated to conte

$sel2:=ds.Employee.query("lastname = Sm@";$querysettings)
$data:=extractData($sel2) // In extractData method the optimization associated to context "shortList" i

$sel3:=ds.Employee.query("lastname = Smith";$querysettings2)
$data:=extractDetailedData($sel3) // In extractDetailedData method an optimization is triggered and ass

$sel4:=ds.Employee.query("lastname = Brown";$querysettings2)
$data:=extractDetailedData($sel4) // In extractDetailedData method the optimization associated to conte

```

## List box basado en una selección de entidades

Entity selection optimization is automatically applied to entity selection-based list boxes in client/server configurations, when displaying and scrolling a list box content: only the attributes displayed in the list box are requested from the server.

A specific "page mode" context is also provided when loading the current entity through the Current item property expression of the list box (see [Collection or entity selection type list boxes](#)). This feature allows you to not overload the list box initial context in this case, especially if the "page" requests additional attributes. Note that only the use of Current item expression will create/use the page context (access through `entitySelection\[index]` will alter the entity selection context).

Subsequent requests to server sent by entity browsing methods will also support this optimization. The following methods automatically associate the optimization context of the source entity to the returned entity:

- `entity.next()`
- `entity.first()`
- `entity.last()`
- `entity.previous()`

For example, the following code loads the selected entity and allows browsing in the entity selection. Entities are loaded in a separate context and the list box initial context is left untouched:

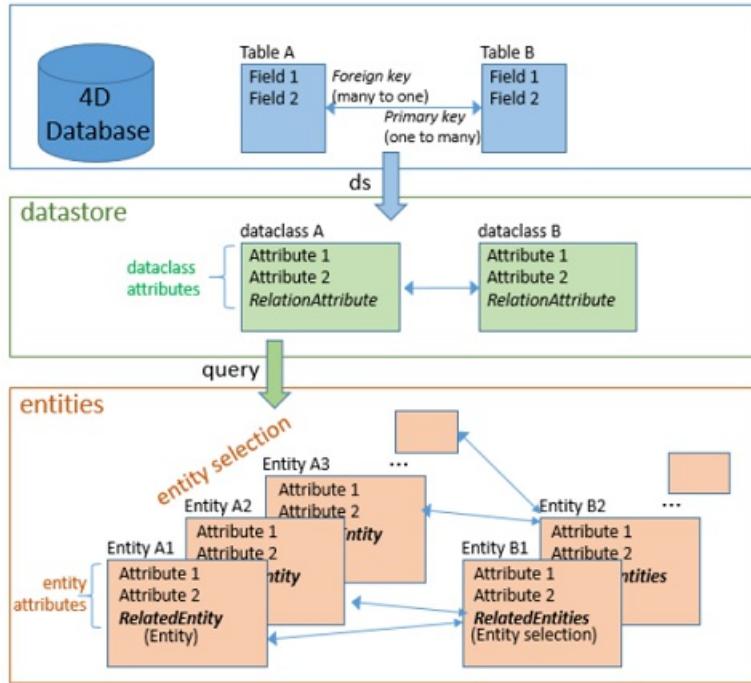
```

$myEntity:=Form.currentElement //current item expression
//... do something
$myEntity:=$myEntity.next() //loads the next entity using the same context

```

# Glosario

## Visión general de los principales conceptos



## Atributo

An attribute is the smallest storage cell in a relational database (see also [Relation attribute](#)). Do not confuse dataclass attributes and entity attributes:

- In a dataclass object, each property is a dataclass attribute that maps to a corresponding field in the corresponding table (same name and type).
- In an entity object, entity attributes are properties that contain values for the corresponding datastore attributes.

Los atributos y las propiedades son conceptos similares. "Atributo" se utiliza para designar las propiedades de la dataclass que almacena datos, mientras que "propiedad" es más genérico y define un dato almacenado dentro de un objeto.

## AttributePath

An attributePath is the path of an attribute inside a given dataclass or entity. Ver también [PropertyPath](#).

## Class code

Código para la(s) función(es) de clase usuarios.

## Atributo calculado

A computed attribute doesn't actually store information. Instead, it determines its value based on other values from the same entity or from other entities, attributes or functions. When a computed attribute is referenced, the underlying "computation" is evaluated to determine the value. Computed attributes may even be assigned values where user-defined code determines what to do during the assignment.

## Data model class

Clase extendida disponible para un objeto del modelo de datos.

## Data model object

Database objects available through the ORDA concept, i.e. datastore, dataclasses, entities and entity selections.

## Data model function

Función de una clase de modelo de datos ORDA.

## Dataclass

A dataclass is an object model that describes the data. Tables in the database provided by the datastore are handled through dataclasses. Each table in the database provided by the datastore has a corresponding dataclass with the same name. Each field of the table is an attribute of the dataclass.

Un dataclass está relacionado con un único datastore.

## DataClass class

Class for specific dataclass objects, in which you can add custom functions.

## Datastore

A datastore is the interface object provided by ORDA to reference a structure and access its data. The main database, returned by the `ds` command, is available as a datastore (the main datastore).

Un datastore ofrece:

- una conexión a la base de datos 4D
- un conjunto de clases de datos para trabajar con la base de datos

The database can be a 4D local database (the Main datastore), or a 4D Server database exposed as REST resource (a Remote datastore).

Un datastore referencia solo a una base de datos. It is, however, possible to open several datastores to access several databases.

## DataStore class

Class for datastore objects, in which you can add custom functions.

## DataStoreImplementation

Internal name of the generic DataStore class in the `4D` class store.

## Copia profunda

A deep copy duplicates an object and all the references it contains. After a deep copy, a copied collection contains duplicated elements and thus, new references, of all of the orginal elements. Ver también Copia superficial.

## ds

`ds` is the 4D language command that returns a [datastore](#) object reference. It matches the datastore available upon the 4D main database.

## Entity

An entity is an object that corresponds to a dataclass model. An entity contains the same attributes as the dataclass.

An entity can be seen as an instance of the dataclass, like a record of the table matching the dataclass in its associated datastore. Sin embargo, una entidad también contiene los datos relacionados. The purpose of the entity is to manage data (create, update, delete).

Para más información, consulte Entidades.

## Entity selection

An entity selection is an object. When querying the datastore, an entity selection is returned. An entity selection is a set of references to entities related to the same dataclass.

Una selección de entidades contiene:

- un conjunto de 0 a X referencias de entidades,
- a length property (always),
- queryPlan and queryPath properties (if asked while querying).

Una selección de entidades también puede estar vacía.

## Generic class

Built-in class for ORDA objects such as entities, or dataclasses. Functions and properties of generic classes are automatically available in user extended classes, e.g. `EmployeeEntity`.

## Lazy loading

Since entities are managed as references, data is loaded only when necessary, i.e. when accessing it in the code or through interface widgets. This optimization principle is called lazy loading.

## Datastore principal

The Datastore object matching the opened 4D database (standalone or client/server). El datastore principal es devuelto por el comando `ds`.

## Método

ORDA objects such as datastores, dataclasses, entity selections, and entities, define classes of objects. They provide specific methods to directly interact with them. Estos métodos también se llaman funciones miembros (member functions). Such methods are used by calling them on an instance of the object.

For example, the `query()` method is a dataclass member function. Si ha almacenado un objeto dataclass en la variable `$myClass`, puede escribir:

```
$myClass.query("name = smith")
```

## Tipo de datos mixtos

In this documentation, "Mixed" data type is used to designate the various type of values that can be stored within dataclass attributes. Incluye:

- number
- texto
- null
- booleano
- fecha
- objeto
- colección
- imagen(\*)

(\*) picture type is not supported by statistical methods such as `entitySelection.max()`.

## Bloqueo optimista

In "optimistic lock" mode, entities are not locked explicitly before updating them. Each entity has an internal stamp that is automatically incremented each time the entity is saved on disk. The `entity.save()` or `entity.drop()` methods will return an error if the stamp of the loaded entity (in memory) and the stamp of the entity on disk do not match, or if the entity has been dropped. Optimistic locking is only available in ORDA implementation. Ver también "Bloqueo pesimista".

## Bloqueo pesimista

A "pessimistic lock" means that an entity is locked prior to its being accessed, using the `entity.lock()` method. Other processes can neither update nor drop the entity until it is unlocked. The classic 4D language only allows pessimistic locks. Ver "Bloqueo optimista".

## Propiedad

Ver [Atributo](#).

*Attributes and properties* are similar concepts. "Atributo" se utiliza para designar las propiedades de la dataclass que almacena datos, mientras que "propiedad" es más genérico y define un dato almacenado dentro de un objeto.

## PropertyPath

A `propertyPath` is the path to a property in a given object. If the property is nested in several levels, each level separated is by a dot (".").

## Regular class

Clase usuario no relacionada a un objeto ORDA.

## Related dataclass

These are dataclasses linked by relation attributes.

## Atributo relacional

Relation attributes are used to conceptualize relations between dataclasses (many-to-one and one-to-many).

- Many-to-one relation (dataclassA references an occurrence of dataclassB): a relation attribute is available in dataclassA and references one instance of dataclassB.
- One-to-many relation (an occurrence of dataclassB references several occurrences of dataclassA): a relation attribute is available in dataclassB and references several instances of dataclassA.

A dataclass can have recursive relation attributes.

In an entity, the value of a relation attribute can be an entity or an entity selection.

## Related entities

A related entity can be seen as the instance of a relation attribute in a dataclass.

Entity selections may refer to related entities according to the relation attributes defined in the corresponding dataclasses.

## Remote datastore

A 4D database opened on a 4D or 4D Server (available through HTTP) and exposed as a REST resource. This database can be referenced locally as a Datastore from other workstations, where it is assigned a locaID. The remote datastore can be used through ORDA concepts (datastore, dataclass, entity selection...). Este uso se somete a un sistema de licencia.

## Sesión

When the 4D application connects to a Remote datastore, a session is created on the 4D Server (HTTP). A session cookie is generated and associated to the local datastore id.

Each time a new session is opened, a license is used. Each time a session is closed, the license is freed.

Inactive sessions are automatically closed after a timeout. The default timeout is 48 hours, it can be set by the developer (it must be  $\geq 60$  minutes).

## Copia superficial (Shallow copy)

A shallow copy only duplicates the structure of elements, and keeps the same internal references. After a shallow copy, two collections will both share the individual elements. Ver también Copia profunda.

## Sello

Utilizado en tecnología de bloqueo "optimista". All entities have an internal counter, the stamp, which is incremented each time the entity is saved. By automatically comparing stamps between an entity being saved and its version stored on disk, 4D can prevent concurrent modifications on the same entities.

## Atributo de almacenamiento

A storage attribute (sometimes referred to as a scalar attribute) is the most basic type of attribute in a datastore class and most directly corresponds to a field in a relational database. A storage attribute holds a single value for each entity in the class.

# Blob

La clase Blob permite crear y manipular los [blob objects](#) ( `4D.Blob` ).

## Resumen

`4D.Blob.new() : 4D.Blob`

`4D.Blob.new( blobScal : Blob ) : 4D.Blob`

`4D.Blob.new( blobObj : 4D.Blob ) : 4D.Blob`

crea un nuevo objeto `4D.Blob` opcionalmente encapsulando una copia de los datos de otro blob (blob escalar o `4D.Blob`)

`.size : Real`

devuelve el tamaño de un `4D.Blob`, expresado en bytes.

`.slice() : 4D.Blob`

`.slice( start : Real ) : 4D.Blob`

`.slice( start : Real; end : Real ) : 4D.Blob`

crea y devuelve un `4D.Blob` que hace referencia a los datos de un subconjunto del blob sobre el que se llama. El blob original no se altera.

## 4D.Blob.new()

► Histórico

`4D.Blob.new() : 4D.Blob`

`4D.Blob.new( blobScal : Blob ) : 4D.Blob`

`4D.Blob.new( blobObj : 4D.Blob ) : 4D.Blob`

Parámetros	Tipo		Descripción
blob	Blob o <code>4D.Blob</code>	->	Blob a copiar
Resultado	<code>4D.Blob</code>	<-	Nuevo <code>4D.Blob</code>

### Descripción

`4D.Blob.new` crea un nuevo objeto `4D.Blob` opcionalmente encapsulando una copia de los datos de otro blob (blob escalar o `4D.Blob`).

Si el parámetro `blob` se omite, el método devuelve un `4D.Blob` vacío.

## .size

`.size : Real`

### Descripción

La propiedad `.size` devuelve el tamaño de un `4D.Blob`, expresado en bytes.

## .slice()

► Histórico

`.slice() : 4D.Blob`

`.slice( start : Real ) : 4D.Blob`

.slice( *start* : Real; *end* : Real ) : 4D.Blob

Parámetros	Tipo		Descripción
start	Real	->	índice del primer byte a incluir en el nuevo 4D.Blob .
end	Real	->	índice del primer byte que no se incluirá en el nuevo 4D.Blob
Resultado	4D.Blob	<-	Nuevo 4D.Blob

## Descripción

.slice() crea y devuelve un 4D.Blob que hace referencia a los datos de un subconjunto del blob sobre el que se llama. El blob original no se altera. El parámetro start es un índice en el blob que indica el primer byte a incluir en el nuevo 4D.Blob . Si indica un valor negativo, 4D lo trata como un desplazamiento desde el final del blob hacia el inicio. Por ejemplo, -10 sería el décimo desde el último byte del blob. El valor por defecto es 0. Si indica un valor de inicio mayor al tamaño del blob fuente, el tamaño del 4D.Blob devuelto es 0, y no contiene datos.

El parámetro end es un índice en el blob que indica el primer byte que no se incluirá en el nuevo 4D.Blob (es decir, el byte situado exactamente en este índice no se incluye). Si indica un valor negativo, 4D lo trata como un desplazamiento desde el final del blob hacia el inicio. Por ejemplo, -10 sería el décimo desde el último byte del blob. El valor por defecto es el tamaño del blob.

## Ejemplo

```
var $myBlob : 4D.Blob

// Almacenar texto en un 4D.Blob
CONVERT FROM TEXT("Hello, World!"; "UTF-8"; $myBlob)
$is4DBlob:=OB Instance of($myBlob; 4D.Blob); //True

$myString:=Convert to text($myBlob; "UTF-8")
// $myString contiene "Hello, World!"

// Crear un nuevo 4D.Blob a partir de $myBlob
$myNewBlob:=$myBlob.slice(0; 5)

$myString:=Convert to text($myNewBlob; "UTF-8")
// $myString contiene "Hello"
```

# Class

Cuando una clase usuario es [definida](#) en el proyecto, se carga en el entorno del lenguaje 4D. Una clase es un objeto en sí mismo, de la clase "Class", que tiene propiedades y una función.

## Resumen

<code>.name : Text</code>	contiene el nombre del objeto <code>4D.Class</code> object
<code>.new( param : any { ;...paramN } ) : 4D.Class</code>	crea y devuelve un objeto <code>cs.className</code> que es una nueva instancia de la clase sobre la que se llama
<code>.superclass : 4D.Class</code>	devuelve la clase padre de la clase

## .name

► Histórico

`.name : Text`

### Descripción

La propiedad `.name` contiene el nombre del objeto `4D.Class` object. Los nombres de clases son sensibles a las mayúsculas y minúsculas.

Esta propiedad es de sólo lectura.

## .new()

► Histórico

`.new( param : any { ;...paramN } ) : 4D.Class`

Parámetros	Tipo		Descripción
param	any	->	Parámetro(s) a pasar a la función constructor
Resultado	4D.Class	<-	Nuevo objeto de la clase

### Descripción

La función `.new()` crea y devuelve un objeto `cs.className` que es una nueva instancia de la clase sobre la que se llama. Esta función está disponible automáticamente en todas las clases del class store `cs`.

Puede pasar uno o más parámetros opcionales `param`, que se pasarán a la función [constructor de la clase](#) (si la hay) en la definición de la clase `className`. Dentro de la función constructor, `This` está ligado al nuevo objeto que se está construyendo.

Si se llama a `.new()` en una clase inexistente, se devuelve un error.

## Ejemplos

Para crear una nueva instancia de la clase Person:

```
var $person : cs.Person  
$person:=cs.Person.new() //crear la nueva instancia  
//$person contiene las funciones de la clase
```

Para crear una nueva instancia de la clase Person con parámetros:

```
//Class: Person.4dm  
Class constructor($firstname : Text; $lastname : Text; $age : Integer)  
    This.firstName:=$firstname  
    This.lastName:=$lastname  
    This.age:=$age
```

```
//En un método  
var $person : cs.Person  
$person:=cs.Person.new("John";"Doe";40)  
//$person.firstName = "John"  
//$person.lastName = "Doe"  
//$person.age = 40
```

## .superclass

► Histórico

.superclass : 4D.Class

### Descripción

La propiedad `.superclass` devuelve la clase padre de la clase. Una superclase puede ser un objeto `4D.Class`, o un objeto `cs.className`. Si la clase no tiene una clase padre, la propiedad devuelve `null`.

Una superclase de clase usuario se declara en una clase utilizando la palabra clave the `Class extends <superclass>`.

Esta propiedad es de sólo lectura.

### Ejemplos

```
$sup:=4D.File.superclass //Document  
$sup:=4D.Document.superclass //Object  
$sup:=4D.Object.superclass //null  
  
// Si creó una clase MyFile  
// with `Class extends File`  
$sup:=cs.MyFile.superclass //File
```

Ver también: [Super](#)

# Collection

La clase Collection gestiona variables de tipo [Collection](#).

Una colección se inicializa con:

New collection {{ ...value : any }} : Collection crea una nueva colección vacía o precargada
New shared collection {{ ...value : any }} : Collection crea una nueva colección compartida vacía o precargada

## Ejemplo

```
var $colVar : Collection //creación de una variable 4D de tipo colección  
$colVar:=New collection //inicialización de la colección y asignación a la variable 4D
```

## Resumen

.average( {propertyPath : Text } ) : Real devuelve la media aritmética (promedio) de los valores definidos en la instancia de la colección
.clear() : Collection elimina todos los elementos de la instancia de la colección y devuelve una colección vacía
.combine( col2 : Collection {; index : Integer } ) : Collection inserta col2 elementos al final o en la posición index especificada en la instancia de la colección y devuelve la colección editada
.concat( value : any { ;...valueN } ) : Collection devuelve una nueva colección que contiene los elementos de la colección original con todos los elementos del parámetro value añadidos al final
.copy() : Collection .copy( option : Integer ) : Collection .copy( option : Integer ; groupWithCol : Collection ) : Collection .copy( option : Integer ; groupWithObj : Object ) : Collection devuelve una copia profunda de la instancia de la colección
.count( { propertyPath : Text } ) : Real devuelve el número de elementos no nulos de la colección
.countValues( value : any {; propertyPath : Text } ) : Real devuelve el número de veces que value está en la colección
.distinct( {option : Integer} ) : Collection .distinct( propertyPath : Text {; option : Integer } ) : Collection devuelve una colección que contiene sólo valores distintos (diferentes) de la colección original
.equal( collection2 : Collection {; option : Integer } ) : Boolean compara collection con collection2

`.every( methodName : Text { ;...param : any } ) : Boolean`

`.every( startFrom : Integer ; methodName : Text { ;...param : any } ) : Boolean`

devuelve true si todos los elementos de la colección han pasado con éxito una prueba implementada en el método *methodName* proporcionado

`.extract( propertyPath : Text { ; option : Integer } ) : Collection`

`.extract( propertyPath : Text ; targetPath : Text { ;...propertyPathN : Text ;... targetPathN : Text } ) : Collection`

crea y devuelve una nueva colección que contiene *propertyPath* valores extraídos de la colección original de objetos

`.fill( value : any ) : Collection`

`.fill( value : any ; startFrom : Integer { ; end : Integer } ) : Collection`

llena la colección con el *value* especificado, opcionalmente desde el índice *startFrom* hasta el índice *end*, y devuelve la colección resultante

`.filter( methodName : Text { ; ...param : any } ) : Collection`

devuelve una nueva colección que contiene los elementos de la colección original para los cuales el resultado del método *methodName* es true

`.find( methodName : Text { ; ...param : any } ) : any`

`.find( startFrom : Integer ; methodName : Text { ; ...param : any } ) : any`

devuelve el primer valor de la colección para el que *methodName*, aplicado a cada elemento, devuelve true

`.findIndex( methodName : Text { ; ...param : any } ) : Integer`

`.findIndex( startFrom : Integer ; methodName : Text { ; ...param : any } ) : Integer`

devuelve el índice, en la colección, del primer valor para el que *methodName*, aplicado a cada elemento, devuelve true

`.indexOf( toSearch : expression { ; startFrom : Integer } ) : Integer`

busca la expresión *toSearch* entre los elementos de la colección y devuelve el índice de la primera ocurrencia encontrada, o -1 si no se encontró

`.indices( queryString : Text { ; ...value : any } ) : Collection`

devuelve los índices, en la colección original, de los elementos de la colección de objetos que coinciden con las condiciones de búsqueda *queryString*, y no los elementos en sí. Los índices se devuelven en orden ascendente.

Esta función no modifica la colección original.

El parámetro *queryString* utiliza la siguiente sintaxis:

```
propertyPath comparator value {logicalOperator propertyPath comparator value}
```

Para una descripción detallada de los parámetros *queryString* y *value*, consulte la función `dataClass.query()`.

Ejemplo

```

var $c; $icol : Collection
$c:=New collection
$c.push(New object("name";"Cleveland";"zc";35049))
$c.push(New object("name";"Blountsville";"zc";35031))

$c.push(New object("name";"Adger";"zc";35006))
$c.push(New object("name";"Clanton";"zc";35046))
$c.push(New object("name";"Clanton";"zc";35045))
$icol:=$c.indices("name = :1";"Cleveland") // $icol=[0]
$icol:=$c.indices("zc > 35040") // $icol=[0,3,4]

```

|

| .insert( *index* : Integer ; *element* : any ) : Collection

inserta *element* en la posición *index* especificada en la instancia de la colección y devuelve la colección editada | | .join( *delimiter* : Text { ; *option* : Integer } ) : Text

convierte todos los elementos de la colección en cadenas y las concatena utilizando la cadena *delimiter* especificada como separador | | .lastIndexOf( *toSearch* : expression { ; *startFrom* : Integer } ) : Integer

busca la expresión *toSearch* entre los elementos de la colección y devuelve el índice de la última ocurrencia | | .length: Integer

devuelve el número de elementos en la colección | | .map( *methodName* : Text { ; ...*param* : any } ) : Collection

crea una nueva colección basada en el resultado de la llamada al método *methodName* sobre cada elemento de la colección original | | .max( { *propertyPath* : Text } ) : any

| | .min( { *propertyPath* : Text } ) : any

devuelve el elemento con el valor más pequeño de la colección | | .orderBy( ) : Collection

.orderBy( *pathStrings* : Text ) : Collection

.orderBy( *pathObjects* : Collection ) : Collection

.orderBy( *ascOrDesc* : Integer ) : Collection

devuelve una nueva colección que contiene todos los elementos de la colección en el orden especificado | | .orderByMethod( *methodName* : Text { ; ...*extraParam* : expression } ) : Collection

devuelve una nueva colección que contiene todos los elementos de la colección en el orden definido a través del método *methodName*.

Esta función devuelve una *copia superficial*, lo que significa que los objetos o colecciones de ambas colecciones comparten la misma referencia. Si la colección original es una colección compartida, la colección devuelta es también una colección compartida.

Esta función no modifica la colección original.

En *methodName*, pase un método de comparación que compare dos valores y devuelva true en *\$1.result* si el primer valor es menor que el segundo. Puede suministrar parámetros adicionales a *methodName* si es necesario.

- *methodName* recibirá los siguientes parámetros:
  - \$1 (objeto), donde:
    - *\$1.value* (todo tipo): primer valor del elemento a comparar
    - *\$1.value2* (todo tipo): segundo valor del elemento a comparar
  - \$2...\$N (cualquier tipo): parámetros adicionales
- *methodName* define el siguiente parámetro:
  - *\$1.result* (boolean): true si *\$1.value < \$1.value2*, false de lo contrario

## Ejemplo 1

Desea ordenar una colección de cadenas en orden numérico en lugar de orden alfabético:

```

var $c; $c2; $c3 : Collection
$c:=New collection
$c.push("33";"4";"1111";"222")
$c2:=$c.orderBy() // $c2=[1111,"222","33","4"], alphabetical order
$c3:=$c.orderByMethod("NumAscending") // $c3=[4,"33","222","1111"]

```

El código para *NumAscending* es:

```
$1.result:=Num($1.value)<Num($1.value2)
```

## Ejemplo 2

Quiere ordenar una colección de cadenas según su longitud:

```

var $fruits; $c2 : Collection
$fruits:=New collection("Orange";"Apple";"Grape";"pear";"Banana";"fig";"Blackberry";"Passion fruit")
$c2:=$fruits.orderByMethod("WordLength")
// $c2=[Passion fruit, Blackberry, Orange, Banana, Apple, Grape, pear, fig]

```

El código para *WordLength* es:

```
$1.result:=Length(String($1.value))>Length(String($1.value2))
```

## Ejemplo 3

Ordenar los elementos de la colección por código de caracteres o alfabéticamente:

```

var $strings1; $strings2 : Collection
$strings1:=New collection("Alpha";"Charlie";"alpha";"bravo";"Bravo";"charlie")

// utilizando el código de caracteres:
$strings2:=$strings1.orderByMethod("sortCollection";sk character codes)
// resultado : ["Alpha", "Bravo", "Charlie", "alpha", "bravo", "charlie"]

// utilizando el lenguaje:
$strings2:=$strings1.orderByMethod("sortCollection";sk strict)
// resultado : ["alpha", "Alpha", "bravo", "Bravo", "charlie", "Charlie"]

```

El método *sortCollection*:

```

var$1Object
var$2Integer // opción de ordenación

$1.result:=(Compare strings($1.value;$1.value2;$2)<0)

```

|| .pop() : any

|| .push( *element* : any { ;...*elementN* } ) : Collection

añade uno o más *elementos* al final de la instancia de la colección y devuelve la colección editada || .query(  
*queryString* : Text ; ...*value* : any ) : Collection  
.query( *queryString* : Text ; *querySettings* : Object ) : Collection

devuelve todos los elementos de una colección de objetos que coinciden con las condiciones de búsqueda || .reduce( *methodName* : Text ) : any

```

.reduce( methodName : Text ; initValue : any { ; ...param : expression } ) : any
|| .remove( index : Integer { ; howMany : Integer } ) : Collection
    elimina uno o más elementos de la posición index especificada en la colección y devuelve la colección editada ||

.resize( size : Integer { ; defaultValue : any } ) : Collection
    ajusta la longitud de la colección al nuevo tamaño especificado y devuelve la colección redimensionada || .reverse( )
    : Collection

    devuelve una copia profunda de la colección con todos sus elementos en orden inverso || .shift() : any

    elimina el primer elemento de la colección y lo devuelve como resultado de la función || .slice( startFrom : Integer
{ ; end : Integer } ) : Collection

    devuelve una porción de una colección en una nueva colección || .some( methodName : Text { ; ...param : any } ) :
Boolean
.some( startFrom : Integer ; methodName : Text { ; ...param : any } ) : Boolean

    devuelve true si al menos un elemento de la colección ha superado con éxito una prueba || .sort( methodName :
Text { ; ...extraParam : any } ) : Collection

    ordena los elementos de la colección original || .sum( { propertyPath : Text } ) : Real

    devuelve la suma de todos los valores de la instancia de la colección || .unshift( value : any { ;...valueN : any } ) :
Collection

    inserta el(es) valor(es) dado(s) al principio de la colección |

```

## Nueva colección

New collection {{ ...*value* : any }} : Collection

Parámetros	Tipo		Descripción
<i>value</i>	Number, Text, Date, Time, Boolean, Object, Collection, Picture, Pointer	->	Valor(es) de collection
Resultado	Collection	<-	Nueva colección

### Descripción

El comando `Nueva colección` crea una nueva colección vacía o precargada y devuelve su referencia.

Si no se pasa ningún parámetro, `New collection` crea una colección vacía y devuelve su referencia.

Debe asignar la referencia devuelta a una variable 4D del tipo Collection.

Tenga en cuenta que las instrucciones `var : Collection` or `C_COLLECTION` declaran una variable de tipo `Collection` pero no crea ninguna colección.

Opcionalmente, puede prellenar la nueva colección pasando uno o varios *valores* como parámetro(s).

De lo contrario, puede añadir o modificar elementos posteriormente por asignación. Por ejemplo:

```
myCol[10]:="My new element"
```

Si el nuevo índice del elemento está más allá del último elemento existente de la colección, la colección se redimensiona automáticamente y a todos los nuevos elementos intermedios se les asigna un valor null.

Puede pasar cualquier número de valores de todos los tipos soportados (number, text, date, picture, pointer, object, collection...). A diferencia de los arrays, las colecciones pueden mezclar datos de diferentes tipos.

Debe prestar atención a los siguientes aspectos de la conversión:

- Si pasa un puntero, se mantiene "tal cual"; se evalúa con el comando `JSON Stringify`
- Las fechas se almacenan como fechas "aaaa-mm-dd" o de cadenas con el formato "AAAA-MM-DDTHH:mm:ss.SSSZ", según la configuración actual "fechas dentro de los objetos" de la base de datos. Al convertir las fechas 4D en texto antes de almacenarlas en la colección, por defecto el programa tiene en cuenta la zona horaria local. Puede modificar este comportamiento utilizando el selector `Dates inside objects` del comando `SET DATABASE PARAMETER`.
- Si pasa un tiempo, se almacena como un número de milisegundos (Real).

### Ejemplo 1

Quiere crear una nueva colección vacía y asignarla a una variable colección 4D:

```
var $myCol : Collection
$myCol:=New collection
//$/myCol=[]
```

### Ejemplo 2

Quiere crear una colección precargada:

```
var $filledColl : Collection
$filledColl:=New collection(33;"mike";"november";->myPtr;Current date)
//$/filledColl=[33,"mike","november",->myPtr,"2017-03-28T22:00:00.000Z"]
```

### Ejemplo 3

Se crea una nueva colección y se añade un nuevo elemento:

```
var $coll : Collection
$coll:=New collection("a";"b";"c")
//$/coll=["a","b","c"]
$coll[9]:="z" //añadir un décimo elemento con valor "z"
$vcollSize:=$coll.length //10
//$/coll=["a","b","c",null,null,null,null,null,null,"z"]
```

## New shared collection

► Histórico

New shared collection `{( ...value : any )} : Collection`

Parámetros	Tipo		Descripción
value	Number, Text, Date, Time, Boolean, Shared object, Shared collection	->	Valor(es) de la collection compartida
Resultado	Collection	<-	New shared collection

### Descripción

El comando `New shared collection` crea una nueva colección compartida vacía o precargada y devuelve su referencia.

La adición de un elemento a esta colección debe estar rodeada por la estructura de uso `Use...End`, de lo contrario se genera un error. Sin embargo, es posible leer un elemento sin estructura.

Para más información sobre las colecciones compartidas, consulte la página [Objetos y colecciones compartidos](#).

Si no se pasa ningún parámetro, `New shared collection` crea una colección compartida vacía y devuelve su referencia.

Debe asignar la referencia devuelta a una variable 4D del tipo Collection.

Tenga en cuenta que las instrucciones `var : Collection` or `C_COLLECTION` declaran una variable de tipo `Collection` pero no crea ninguna colección.

Opcionalmente, puede prellenar la nueva colección compartida pasando uno o varios *valores* como parámetro(s). De lo contrario, puede añadir o modificar elementos posteriormente por asignación notación objeto (ver ejemplo).

Si el nuevo índice del elemento está más allá del último elemento existente de la colección compartida, la colección se redimensiona automáticamente y a todos los nuevos elementos intermedios se les asigna un valor null.

Puede pasar cualquier número de valores de los siguientes tipos soportados:

- number (real, longint...). Los valores numéricos se almacenan siempre como reales.
- texto
- booleano
- fecha
- time (almacenado como número de milisegundos - real)
- null
- objeto compartido(\*)
- collection compartida(\*)

A diferencia de las colecciones estándar (no compartidas), las colecciones compartidas no soportan imágenes, punteros y objetos o colecciones no compartidas.

(\*)Cuando un objeto o colección compartida se añade a una colección compartida, comparten el mismo *identificador de bloqueo*. Para obtener más información sobre este punto, consulte la guía del Desarrollador 4D.

## Ejemplo

```
$mySharedCol:=New shared collection("alpha";"omega")
Use($mySharedCol)
  $mySharedCol[1]:="beta"
End use
```

## .average()

► Histórico

`.average( {propertyPath : Text} ) : Real`

Parámetros	Tipo		Descripción
propertyPath	Texto	->	Ruta de la propiedad del objeto que se utilizará para el cálculo
Resultado	Real, Undefined	<-	Media aritmética (promedio) de los valores de la colección

## Descripción

La función `.average()` devuelve la media aritmética (promedio) de los valores definidos en la instancia de la colección.

Para el cálculo sólo se tienen en cuenta los elementos numéricos (se ignoran otros tipos de elementos).

Si la colección contiene objetos, pasa el parámetro `propertyPath` para indicar la propiedad del objeto a tener en cuenta.

`.average()` devuelve `undefined` if:

- la colección está vacía,

- la colección no contiene elementos numéricos,
- *propertyPath* no se encuentra en la colección.

## Ejemplo 1

```
var $col : Collection
$col:=New collection(10;20;"Monday";True;6)
$vAvg:=$col.average() //12
```

## Ejemplo 2

```
var $col : Collection
$col:=New collection
$col.push(New object("name";"Smith";"salary";10000))
$col.push(New object("name";"Wesson";"salary";50000))
$col.push(New object("name";"Gross";"salary";10500))
$vAvg:=$col.average("salary") //23500
```

## .clear()

► Histórico

.clear() : Collection

Parámetros	Tipo		Descripción
Resultado	Collection	<-	Colección original con todos los elementos eliminados

### Descripción

La función `.clear()` elimina todos los elementos de la instancia de la colección y devuelve una colección vacía.

Esta función modifica la colección original.

## Ejemplo

```
var $col : Collection
$col:=New collection(1;2;5)
$col.clear()
$vSize:=$col.length //vSize=0
```

## .combine()

► Histórico

.combine( *col2* : Collection {; *index* : Integer } ) : Collection

Parámetros	Tipo		Descripción
<i>col2</i>	Collection	->	Colección a combinar
<i>index</i>	Integer	->	Posición a la que se deben combinar los elementos de inserción en la colección (por defecto=length+1)
Resultado	Collection	<-	Colección original que contiene elementos combinados

### Descripción

La función `.combine()` inserta `col/2` elementos al final o en la posición `index` especificada en la instancia de la colección y devuelve la colección editada. A diferencia de la función `.insert()`, `.combine()` añade cada valor de `col/2` en la colección original, y no como un solo elemento de la colección.

Esta función modifica la colección original.

Por defecto, los elementos `col/2` se añaden al final de la colección original. Puede pasar en `index` la posición en la que quiere que se inserten los elementos `col/2` en la colección.

Atención: recuerde que los elementos de la colección están numerados desde 0.

- Si `índice >` la longitud de la colección, el `índice` inicial real se fijará en la longitud de la colección.
- Si `índice < 0`, se recalcula como `index:=index+length` (se considera el desplazamiento desde el final de la colección).
- Si el valor calculado es negativo, `index` toma el valor 0.

## Ejemplo

```
var $c; $fruits : Collection  
$c:=New collection(1;2;3;4;5;6)  
$fruits:=New collection("Orange";"Banana";"Apple";"Grape")  
$c.combine($fruits;3) // [1,2,3,"Orange","Banana","Apple","Grape",4,5,6]
```

## .concat()

► Histórico

`.concat( value : any { ;...valueN } ) : Collection`

Parámetros	Tipo		Descripción
value	Number, Text, Object, Collection, Date, Time, Boolean, Picture	->	Valores a concatenar. Si <code>value</code> es una colección, todos los elementos de la colección se añaden a la colección original
Resultado	Collection	<-	Nueva colección con valor(es) añadido(s) a la colección original

## Descripción

La función `.concat()` devuelve una nueva colección que contiene los elementos de la colección original con todos los elementos del parámetro `value` añadidos al final.

Esta función no modifica la colección original.

Si `value` es una colección, todos sus elementos se añaden al final de la colección original. Si `value` no es una colección, se añade ella misma como un nuevo elemento.

## Ejemplo

```
var $c : Collection  
$c:=New collection(1;2;3;4;5)  
$fruits:=New collection("Orange";"Banana";"Apple";"Grape")  
$fruits.push(New object("Intruder";"Tomato"))  
$c2:=$c.concat($fruits) // [1,2,3,4,5,"Orange","Banana","Apple","Grape",{"Intruder":"Tomato"}]  
$c2:=$c.concat(6;7;8) // [1,2,3,4,5,6,7,8]
```

## .copy()

► Histórico

.copy() : Collection

.copy( option : Integer ) : Collection

.copy( option : Integer ; groupWithCol : Collection ) : Collection

.copy( option : Integer ; groupWithObj : Object ) : Collection

Parámetros	Tipo		Descripción
option	Integer	->	ck resolve pointers : resolver punteros antes de copiar, ck shared : devolver una colección compartida
groupWithCol	Collection	->	Colección compartida que se agrupa con la colección resultante
groupWithObj	Objeto	->	Objeto compartido que se agrupa con la colección resultante
Resultado	Collection	<-	Copia profunda de la colección original

### Descripción

La función `.copy()` devuelve una copia profunda de la instancia de la colección. *Deep copy* significa que los objetos o colecciones dentro de la colección original se duplican y no comparten ninguna referencia con la colección devuelta.

Esta función no modifica la colección original.

Si se pasa, el parámetro *opción* puede contener una de las siguientes constantes (o ambas):

option	Descripción
ck resolve pointers	Si la colección original contiene valores de tipo puntero, por defecto la copia también contiene los punteros. Sin embargo, puede resolver los punteros al copiar pasando ck resolve pointers. En este caso, cada puntero presente en la colección se evalúa al copiar y se utiliza su valor desreferenciado.
ck shared	Por defecto, copy() devuelve una colección Clásica (no compartida), incluso si el comando se aplica a una colección compartida. Pasa la constante compartida ck para crear una colección compartida. En este caso, puede utilizar el parámetro groupWith para asociar la colección compartida con otra colección u objeto (ver más adelante).

Los parámetros *groupWithCol* o *groupWithObj* permiten designar una colección o un objeto al que se debe asociar la colección resultante.

### Ejemplo 1

Queremos copiar la colección regular (no compartida) `$lastnames` en el objeto compartido `$sharedObject`. Para ello, debemos crear una copia compartida de la colección (`$sharedLastnames`).

```
var $sharedObject : Object
var $lastnames;$sharedLastnames : Collection
var $text : Text

$sharedObject:=New shared object

$text:=Document to text(Get 4D folder(Current resources folder)+"lastnames.txt")
$lastnames:=JSON Parse($text) //lastnames es una colección estándar

$sharedLastnames:=$lastnames.copy(ck shared) //sharedLastnames es una colección compartida

//Now we can put $sharedLastnames into $sharedObject
Use($sharedObject)
    $sharedObject.lastnames:=$sharedLastnames
End use
```

## Ejemplo 2

Queremos combinar `$sharedColl1` and `$sharedColl2`. Dado que pertenecen a diferentes grupos compartidos, una combinación directa daría lugar a un error. Por lo tanto, debemos hacer una copia compartida de `$sharedColl1` y designar `$sharedColl2` como grupo compartido para la copia.

```
var $sharedColl1;$sharedColl2;$copyColl : Collection  
  
$sharedColl1:=New shared collection(New shared object("lastname";"Smith"))  
$sharedColl2:=New shared collection(New shared object("lastname";"Brown"))  
  
//copyColl pertenece al mismo grupo compartido que $sharedColl2  
$copyColl:=$sharedColl1.copy(ck shared;$sharedColl2)  
Use($sharedColl2)  
    $sharedColl2.combine($copyColl)  
End use
```

## Ejemplo 3

Tenemos una colección Clásica (`$lastnames`) y queremos ponerla en el Storage de la aplicación. Para ello, debemos crear previamente una copia compartida (`$sharedLastnames`).

```
var $lastnames;$sharedLastnames : Collection  
var $text : Text  
  
$text:=Document to text(Get 4D folder(Current resources folder)+"lastnames.txt")  
$lastnames:=JSON Parse($text) //lastnames es una colección clásica  
  
$sharedLastnames:=$lastnames.copy(ck shared) // copia compartida  
  
Use(Storage)  
    Storage.lastnames:=$sharedLastnames  
End use
```

## Ejemplo 4

Este ejemplo ilustra el uso de la opción `ck resolve pointers`:

```
var $col : Collection  
var $p : Pointer  
$p:==>$what  
  
$col:=New collection  
$col.push(New object("alpha";"Hello";"num";1))  
$col.push(New object("beta";"You";"what";$p))  
  
$col2:=$col.copy()  
$col2[1].beta:="World!"  
ALERT($col[0].alpha+" "+$col2[1].beta) //muestra "Hello World!"  
  
$what:="You!"  
$col3:=$col2.copy(ck resolve pointers)  
ALERT($col3[0].alpha+" "+$col3[1].what) //muestra "Hello You!"
```

## .count()

► Histórico

`.count( { propertyPath : Text } ) : Real`

Parámetros	Tipo		Descripción
propertyPath	Texto	->	Ruta de la propiedad del objeto que se utilizará para el cálculo
Resultado	Real	<-	Número de elementos en la colección

## Descripción

La función `.count()` devuelve el número de elementos no nulos de la colección.

Si la colección contiene objetos, se puede pasar el parámetro *propertyPath*. En este caso, sólo se tienen en cuenta los elementos que contienen la *propertyPath*.

## Ejemplo

```
var $col : Collection
var $count1;$count2 : Real
$col:=New collection(20;30;Null;40)
$col.push(New object("name";"Smith";"salary";10000))
$col.push(New object("name";"Wesson";"salary";50000))
$col.push(New object("name";"Gross";"salary";10500))
$col.push(New object("lastName";"Henry";"salary";12000))
$count1:=$col.count() // $count1=7
$count2:=$col.count("name") // $count2=3
```

## .countValues()

► Histórico

`.countValues( value : any {; propertyPath : Text } ) : Real`

Parámetros	Tipo		Descripción
value	Text, Number, Boolean, Date, Object, Collection	->	Valor a contar
propertyPath	Texto	->	Ruta de la propiedad del objeto que se utilizará para el cálculo
Resultado	Real	<-	Número de ocurrencias del valor

## Descripción

La función `.countValues()` devuelve el número de veces que *value* está en la colección.

Puede pasar en *value*:

- un valor escalar (texto, número, booleano, fecha),
- una referencia de objeto o de colección.

Para encontrar un elemento, el tipo de *value* debe ser equivalente al tipo del elemento; el método utiliza el operador de igualdad.

El parámetro opcional *propertyPath* permite contar valores dentro de una colección de objetos: pase en *propertyPath* la ruta de la propiedad cuyos valores quiere contar.

Esta función no modifica la colección original.

## Ejemplo 1

```

var $col : Collection
var $vCount : Integer
$col:=New collection(1;2;5;5;5;3;6;4)
$vCount:=$col.countValues(5) // $vCount=3

```

## Ejemplo 2

```

var $col : Collection
var $vCount : Integer
$col:=New collection
$col.push(New object("name";"Smith";"age";5))
$col.push(New object("name";"Wesson";"age";2))
$col.push(New object("name";"Jones";"age";3))
$col.push(New object("name";"Henry";"age";4))
$col.push(New object("name";"Gross";"age";5))
$vCount:=$col.countValues(5;"age") // $vCount=2

```

## Ejemplo 3

```

var $numbers; $letters : Collection
var $vCount : Integer

$letters:=New collection("a";"b";"c")
$numbers:=New collection(1;2;$letters;3;4;5)

$vCount:=$numbers.countValues($letters) // $vCount=1

```

## .distinct()

► Histórico

`.distinct( {option : Integer} ) : Collection`  
`.distinct( propertyPath : Text {; option : Integer } ) : Collection`

Parámetros	Tipo		Descripción
option	Integer	->	<code>ck diacritical</code> : evaluación diacrítica ("A" # "a" por ejemplo)
propertyPath	Texto	->	Ruta del atributo cuyos valores distintos desea obtener
Resultado	Collection	<-	Nueva colección con sólo valores distintos

### Descripción

La función `.distinct()` devuelve una colección que contiene sólo valores distintos (diferentes) de la colección original.

Esta función no modifica la colección original.

La colección devuelta se clasifica automáticamente. Los valores Null no se devuelven.

Por defecto, se realiza una evaluación no diacrítica. Si desea que la evaluación distinga entre mayúsculas y minúsculas o que diferencie los caracteres acentuados, pase la constante `ck diacritical` en el parámetro `option`.

Si la colección contiene objetos, puede pasar el parámetro `propertyPath` para indicar la propiedad del objeto cuyos valores distintos desea obtener.

## Ejemplo

```

var $c; $c2 : Collection
$c:=New collection
$c.push("a";"b";"c";"A";"B";"c";"b";"b")
$c.push(New object("size";1))
$c.push(New object("size";3))
$c.push(New object("size";1))
$c2:=$c.distinct() // $c2=[{"a":1,"b":1,"c":1,"size":1}, {"a":2,"b":2,"c":2,"size":3}, {"a":3,"b":3,"c":3,"size":1}]
$c2:=$c.distinct(ck diacritical) // $c2=[{"a":1,"A":1,"B":2,"c":1,"size":1}, {"a":2,"b":2,"c":2,"size":3}, {"a":3,"b":3,"c":3,"size":1}]
$c2:=$c.distinct("size") // $c2=[1,3]

```

## .equal()

► Histórico

.equal( collection2 : Collection {; option : Integer } ) : Boolean

Parámetros	Tipo		Descripción
collection2	Collection	->	Colección a comparar
option	Integer	->	ck diacritical : evaluación diacrítica ("A" # "a" por ejemplo)
Resultado	Booleano	<-	True si las colecciones son idénticas, false en caso contrario

### Descripción

La función `.equal()` compara collection con collection2 y devuelve true si son idénticas (comparación profunda深深 comparison).

Por defecto, se realiza una evaluación no diacrítica. Si desea que la evaluación diferencie entre mayúsculas y minúsculas o que diferencie los caracteres acentuados, pase la constante `ck diacritical` en el parámetro option.

Los elementos con valores Null no son iguales a los elementos Undefined.

### Ejemplo

```

var $c; $c2 : Collection
var $b : Boolean

$c:=New collection(New object("a";1;"b";"orange");2;3)
$c2:=New collection(New object("a";1;"b";"orange");2;3;4)
$b:=$c.equal($c2) // false

$c:=New collection(New object("1";"a";"b";"orange");2;3)
$c2:=New collection(New object("a";1;"b";"orange");2;3)
$b:=$c.equal($c2) // false

$c:=New collection(New object("a";1;"b";"orange");2;3)
$c2:=New collection(New object("a";1;"b";"0Range");2;3)
$b:=$c.equal($c2) // true

$c:=New collection(New object("a";1;"b";"orange");2;3)
$c2:=New collection(New object("a";1;"b";"0Range");2;3)
$b:=$c.equal($c2;ck diacritical) //false

```

## .every()

► Histórico

.every( methodName : Text { ;...param : any } ) : Boolean

.every( *startFrom* : Integer ; *methodName* : Text { ;...*param* : any } ) : Boolean

Parámetros	Tipo		Descripción
<i>startFrom</i>	Integer	->	Índice para iniciar la prueba en
<i>methodName</i>	Texto	->	Nombre del método a llamar para la prueba
<i>param</i>	Mixed	->	Parámetro(s) a pasar a <i>methodName</i>
Resultado	Booleano	<-	True si todos los elementos han pasado la prueba con éxito

## Descripción

La función `.every()` devuelve true si todos los elementos de la colección han pasado con éxito una prueba implementada en el método *methodName* proporcionado.

En *methodName*, pase el nombre del método a utilizar para evaluar los elementos de la colección, junto con su(s) parámetro(s) en *param* (opcional). *methodName* puede realizar cualquier prueba, con o sin los parámetros. Este método recibe un `Object` en el primer parámetro (\$1) y debe definir `$1.result` como true para cada elemento que cumpla la prueba.

*methodName* recibe los siguientes parámetros:

- en `$1.value`: valor del elemento a evaluar
- en `$2`: param
- en `$N...`: paramN...

*methodName* define el(los) siguiente(s) parámetro(s):

- `>$1.result` (Boolean): true si la evaluación del valor del elemento tiene éxito, si no false.
- `$1.stop` (Boolean, opcional): true para detener la retrollamada del método. El valor devuelto es el último calculado.

En todos los casos, en el momento en que la función `.every()` encuentre el primer elemento de la colección que devuelva false en `$1.result`, deja de llamar a *methodName* y devuelve false.

Por defecto, `.every()` comprueba toda la colección. Opcionalmente, se puede pasar en *startFrom* el índice del elemento desde el que iniciar la prueba.

- Si `startFrom >=` la longitud de la colección, se devuelve false, lo que significa que la colección no se prueba.
- Si `startFrom < 0`, se considera el desplazamiento desde el final de la colección (`startFrom:=startFrom+length`).
- Si `startFrom = 0`, se busca en toda la colección (por defecto).

## Ejemplo 1

```
var $c : Collection
var $b : Boolean
$c:=New collection
$c.push(5;3;1;4;6;2)
$b:=$c.every("NumberGreaterThan0") //devuelve true
$c.push(-1)
$b:=$c.every("NumberGreaterThan0") //devuelve false
```

Con lo siguiente método *NumberGreaterThan0*:

```
$1.result:=$1.value>0
```

## Ejemplo 2

Este ejemplo comprueba que todos los elementos de una colección son de tipo real:

```

var $c : Collection
var $b : Boolean
$c:=New collection
$c.push(5;3;1;4;6;2)
$b:=$c.every("TypeLookUp";Is real) //b=true
$c:=$c.push(New object("name";"Cleveland";"zc";35049))
$c:=$c.push(New object("name";"Blountsville";"zc";35031))
$b:=$c.every("TypeLookUp";Is real) //b=false

```

Con lo siguiente método *TypeLookUp*:

```

#DECLARE ($toEval : Object ; $param : Integer) //1; $2
If(Value type($toEval.value)=$param)
    $toEval.result:=True
End if

```

## .extract()

► Histórico

.extract( *propertyPath* : Text { ; *option* : Integer } ) : Collection

.extract( *propertyPath* : Text ; *targetPath* : Text { ;...*propertyPathN* : Text ;... *targetPathN* : Text } ) : Collection

Parámetros	Tipo		Descripción
<i>propertyPath</i>	Texto	->	Ruta de la propiedad del objeto cuyos valores deben ser extraídos a la nueva colección
<i>targetpath</i>	Texto	->	Ruta de la propiedad de destino o nombre de la propiedad
<i>option</i>	Integer	->	ck keep null : incluye la propiedad null en la colección devuelta (se ignora por defecto). Parámetro ignorado si se pasa <i>targetPath</i> .
Resultado	Collection	<-	Nueva colección que contiene los valores extraídos

### Descripción

La función `.extract()` crea y devuelve una nueva colección que contiene *propertyPath* valores extraídos de la colección original de objetos.

Esta función no modifica la colección original.

El contenido de la colección devuelta depende del parámetro *targetPath*:

- Si se omite el parámetro, *targetPath*, `.extract()` rellena la nueva colección con los valores de *propertyPath* de la colección original.

Por defecto, los elementos cuya *propertyPath* es null o indefinida se ignoran en la colección resultante. Puede pasar la constante `ck keep null` en el parámetro *option* para incluir estos valores como elementos nulos en la colección devuelta.

- Si se pasan uno o más parámetros *targetPath*, `.extract()` rellena la nueva colección con las propiedades *propertyPath* y cada elemento de la nueva colección es un objeto con propiedades *targetPath* llenadas con las propiedades *propertyPath* correspondientes. Se mantienen los valores null (el parámetro *option* se ignora) con esta sintaxis.

### Ejemplo 1

```

var $c : Collection
$c:=New collection
$c.push(New object("name";"Cleveland"))
$c.push(New object("zip";5321))
$c.push(New object("name";"Blountsville"))
$c.push(42)
$c2:=$c.extract("name") // $c2=[Cleveland,Blountsville]
$c2:=$c.extract("name";ck keep null) // $c2=[Cleveland,null,Blountsville,null]

```

## Ejemplo 2

```

var $c : Collection
$c:=New collection
$c.push(New object("zc";35060))
$c.push(New object("name";Null;"zc";35049))
$c.push(New object("name";"Cleveland";"zc";35049))
$c.push(New object("name";"Blountsville";"zc";35031))
$c.push(New object("name";"Adger";"zc";35006))
$c.push(New object("name";"Clanton";"zc";35046))
$c.push(New object("name";"Clanton";"zc";35045))
$c2:=$c.extract("name";"City") // $c2=[{City:null},{City:Cleveland},{City:Blountsville},{City:Adger},{City:Clanton},{City:Clanton}]
$c2:=$c.extract("name";"City";"zc";"Zip") // $c2=[{Zip:35060},{City:null,Zip:35049},{City:Cleveland,Zip:35031},{City:Adger,Zip:35006},{City:Clanton,Zip:35046},{City:Clanton,Zip:35045}]

```

## .fill()

► Histórico

`.fill( value : any ) : Collection`  
`.fill( value : any ; startFrom : Integer { ; end : Integer } ) : Collection`

Parámetros	Tipo		Descripción
value	number, Text, Collection, Object, Date, Boolean	->	Valor a asignar
startFrom	Integer	->	Índice de inicio (incluido)
end	Integer	->	Índice final (no incluido)
Resultado	colección	<-	Colección original con valores rellenados

### Descripción

La función `.fill()` llena la colección con el `value` especificado, opcionalmente desde el índice `startFrom` hasta el índice `end`, y devuelve la colección resultante.

Esta función modifica la colección original.

- Si se omite el parámetro `startFrom`, `value` se define en todos los elementos de la colección (`startFrom=0`).
- Si se pasa el parámetro `startFrom` y se omite `end`, `value` se establece en los elementos de la colección empezando por `startFrom` hasta el último elemento de la colección (`end=length`).
- Si se pasan ambos parámetros `startFrom` y `end`, `value` se establece en los elementos de la colección empezando por `startFrom` hasta el elemento `end`.

En caso de incoherencia, se aplican las siguientes reglas:

- Si `startFrom < 0`, se recalcula como `startFrom:=startFrom+length` (se considera el desplazamiento desde el final de la colección). Si el valor calculado es negativo, `startFrom` toma el valor 0.
- Si `end < 0`, se recalcula como `end:=end+length`.
- Si `end < startFrom` (valores pasados o calculados), el método no hace nada.

## Ejemplo

```
var $c : Collection
$c:=New collection(1;2;3;"Lemon";Null;"";4;5)
$c.fill("2") // $c:=[2,2,2,2,2,2,2]
$c.fill("Hello";5) // $c=[2,2,2,2,Hello,Hello,Hello]
$c.fill(0;1;5) // $c=[2,0,0,0,0>Hello,Hello,Hello]
$c.fill("world";1;-5) // -5+8=3 -> $c=[2,"world","world",0,0>Hello,Hello,Hello]
```

## .filter()

► Histórico

.filter( *methodName* : Text { ; ...*param* : any } ) : Collection

Parámetros	Tipo		Descripción
methodName	Texto	->	Nombre de la función a la que se llama para filtrar la colección
param	Mixed	->	Parámetro(s) a pasar a <i>methodName</i>
Resultado	Collection	<-	Nueva colección que contiene elementos filtrados (copia superficial)

### Descripción

La función `.filter()` devuelve una nueva colección que contiene los elementos de la colección original para los cuales el resultado del método *methodName* es true. Esta función devuelve una *copia superficial*, lo que significa que los objetos o colecciones de ambas colecciones comparten la misma referencia. Si la colección original es una colección compartida, la colección devuelta es también una colección compartida.

Esta función no modifica la colección original.

En *methodName*, pase el nombre del método a utilizar para evaluar los elementos de la colección, junto con su(s) parámetro(s) en *param* (opcional). *methodName* puede realizar cualquier prueba, con o sin los parámetros. En *methodName*, pase el nombre del método a utilizar para evaluar los elementos de la colección, junto con su(s) parámetro(s) en *param* (opcional).

*methodName* recibe los siguientes parámetros:

- en `$1.value`: valor del elemento a filtrar
- en `$2: param`
- en `$N...: param2...paramN`

*methodName* define el(los) siguiente(s) parámetro(s):

- `$1.result` (boolean): true si el valor del elemento coincide con la condición del filtro y debe mantenerse.
- `$1.stop` (boolean, opcional): true para detener la retrollamada del método. El valor devuelto es el último calculado.

### Ejemplo 1

Desea obtener la colección de elementos de texto cuya longitud es inferior a 6:

```
var $col;$colNew : Collection
$c:=New collection("hello";"world";"red horse";66;"tim";"san jose";"miami")
$cNew:=$col.filter("LengthLessThan";6)
//$colNew=["hello","world","tim","miami"]
```

El código del método `LengthLessThan` es:

```

C_OBJECT($1)
C_LONGINT($2)
If(Value_type($1.value)=Is text)
    $1.result:=(Length($1.value))<$2
End if

```

## Ejemplo 2

Quiere filtrar los elementos según su tipo de valor:

```

var $c;$c2;$c3 : Collection
$c:=New collection(5;3;1;4;6;2)
$c.push(New object("name";"Cleveland";"zc";35049))
$c.push(New object("name";"Blountsville";"zc";35031))
$c2:=$c.filter("TypeLookUp";Is real) // $c2=[5,3,1,4,6,2]
$c3:=$c.filter("TypeLookUp";Is object)
// $c3=[{name:Cleveland,zc:35049},{name:Blountsville,zc:35031}]

```

El código de *TypeLookUp* es:

```

C_OBJECT($1)
C_LONGINT($2)
If(0B Get type($1;"value")=$2)

    $1.result:=True
End if

```

## .find()

► Histórico

.find( *methodName* : Text { ; ...*param* : any } ) : any  
 .find( *startFrom* : Integer ; *methodName* : Text { ; ...*param* : any } ) : any

Parámetros	Tipo		Descripción
<i>startFrom</i>	Integer	->	Índice para iniciar la búsqueda en
<i>methodName</i>	Texto	->	Nombre de la función a la que se llama para la búsqueda
<i>param</i>	any	->	Parámetro(s) a pasar a <i>methodName</i>
Resultado	any	<-	Primer valor encontrado, o Undefined si no se encuentra

### Descripción

La función `.find()` devuelve el primer valor de la colección para el que *methodName*, aplicado a cada elemento, devuelve true.

Esta función no modifica la colección original.

En *methodName*, pase el nombre del método a utilizar para evaluar los elementos de la colección, junto con su(s) parámetro(s) en *param* (opcional). *methodName* puede realizar cualquier prueba, con o sin los parámetros. En *methodName*, pase el nombre del método a utilizar para evaluar los elementos de la colección, junto con su(s) parámetro(s) en *param* (opcional).

*methodName* recibe los siguientes parámetros:

- en `$1.value`: valor del elemento a evaluar

- en \$2: param
- en \$N...: param2...paramN

*methodName* define el(los) siguiente(s) parámetro(s):

- >\$1.result (boolean): true si el valor del elemento coincide con la condición de búsqueda.
- \$1.stop (boolean, opcional): true para detener la retrollamada del método. El valor devuelto es el último calculado.

Por defecto, `.find()` busca en toda la colección. Opcionalmente, se puede pasar en *startFrom* el índice del elemento desde el que iniciar la búsqueda.

- Si *startFrom*  $\geq$  la longitud de la colección, se devuelve -1, lo que significa que la colección no se busca.
- Si *startFrom*  $< 0$ , se considera el desplazamiento desde el final de la colección (*startFrom:=startFrom+length*).  
Nota: incluso si *startFrom* es negativo, la colección se sigue buscando de izquierda a derecha.
- Si *startFrom* = 0, se busca en toda la colección (por defecto).

## Ejemplo 1

Quiere obtener el primer elemento con una longitud menor que 5:

```
var $col : Collection
$col:=New collection("hello";"world";4;"red horse";"tim";"san jose")
$value:=$col.find("LengthLessThan";5) //$/value="tim"
```

El código del método *LengthLessThan* es:

```
var $1 : Object
var $2 : Integer
If(Value type($1.value)=Is text)
    $1.result:=(Length($1.value))<$2
End if
```

## Ejemplo 2

Quiere encontrar el nombre de una ciudad dentro de una colección:

```
var $c : Collection
var $c2 : Object
$c:=New collection
$c.push(New object("name";"Cleveland";"zc";35049))
$c.push(New object("name";"Blountsville";"zc";35031))
$c.push(New object("name";"Adger";"zc";35006))
$c.push(New object("name";"Clanton";"zc";35046))
$c.push(New object("name";"Clanton";"zc";35045))
$c2:=$c.find("FindCity";"Clanton") //$/c2={name:Clanton,zc:35046}
```

El código para *FindCity* es:

```
var $1 : Object
var $2 : Text
$1.result:=$1.value.name=$2 //name es un nombre de propiedad de los objetos de la colección
```

## .findIndex()

► Histórico

```
.findIndex( methodName : Text { ; ...param : any } ) : Integer
.findIndex( startFrom : Integer ; methodName : Text { ; ...param : any } ) : Integer
```

Parámetros	Tipo		Descripción
startFrom	Integer	->	Índice para iniciar la búsqueda en
methodName	Texto	->	Nombre de la función a la que se llama para la búsqueda
param	any	->	Parámetro(s) a pasar a <i>methodName</i>
Resultado	Integer	<-	Índice del primer valor encontrado, o -1 si no se encuentra

## Descripción

La función `.findIndex()` devuelve el índice, en la colección, del primer valor para el que *methodName*, aplicado a cada elemento, devuelve true.

Esta función no modifica la colección original.

En *methodName*, pase el nombre del método a utilizar para evaluar los elementos de la colección, junto con su(s) parámetro(s) en *param* (opcional). En *methodName*, pase el nombre del método a utilizar para evaluar los elementos de la colección, junto con su(s) parámetro(s) en *param* (opcional). *methodName* puede realizar cualquier prueba, con o sin los parámetros.

*methodName* recibe los siguientes parámetros:

- en `$1.value`: valor del elemento a evaluar
- en `$2: param`
- en `$N...: param2...paramN`

*methodName* define el(los) siguiente(s) parámetro(s):

- `>$1.result` (boolean): true si el valor del elemento coincide con la condición de búsqueda.
- `$1.stop` (boolean, opcional): true para detener la retrollamada del método. El valor devuelto es el último calculado.

Por defecto, `.findIndex()` busca en toda la colección. Opcionalmente, se puede pasar en *startFrom* el índice del elemento desde el que iniciar la búsqueda.

- Si *startFrom*  $\geq$  la longitud de la colección, se devuelve -1, lo que significa que la colección no se busca.
- Si *startFrom*  $< 0$ , se considera el desplazamiento desde el final de la colección (*startFrom:=startFrom+length*). Nota: incluso si *startFrom* es negativo, la colección se sigue buscando de izquierda a derecha.
- Si *startFrom* = 0, se busca en toda la colección (por defecto).

## Ejemplo

Quiere encontrar la posición del primer nombre de ciudad dentro de una colección:

```
var $c : Collection
var $val2;$val3 : Integer
$c:=New collection
$c.push(New object("name";"Cleveland";"zc";35049))
$c.push(New object("name";"Blountsville";"zc";35031))
$c.push(New object("name";"Adger";"zc";35006))
$c.push(New object("name";"Clanton";"zc";35046))
$c.push(New object("name";"Clanton";"zc";35045))
$val2:=$c.findIndex("FindCity";"Clanton") // $val2=3
$val3:=$c.findIndex($val2+1;"FindCity";"Clanton") // $val3=4
```

El código del método *FindCity* es:

```
var $1 : Object
var $2 : Text
$1.result:=$1.value.name=$2
```

## .indexOf()

► Histórico

.indexOf( *toSearch* : expression { ; *startFrom* : Integer } ) : Integer

Parámetros	Tipo		Descripción
toSearch	expresión	->	Expresión a buscar en la colección
startFrom	Integer	->	Índice para iniciar la búsqueda en
Resultado	Integer	<-	Índice de la primera ocurrencia de <i>toSearch</i> en la colección, -1 si no se encuentra

### Descripción

La función `.indexOf()` busca la expresión *toSearch* entre los elementos de la colección y devuelve el índice de la primera ocurrencia encontrada, o -1 si no se encontró.

Esta función no modifica la colección original.

En *toSearch*, pase la expresión a encontrar en la colección. Puede pasar:

- un valor escalar (texto, número, booleano, fecha),
- el valor null,
- una referencia de objeto o de colección.

*toSearch* debe coincidir exactamente con el elemento a encontrar (se aplican las mismas reglas que para el operador de igualdad del tipo de datos).

Opcionalmente, puede pasar el índice de la colección desde el que iniciar la búsqueda en *startFrom*.

- Si *startFrom*  $\geq$  la longitud de la colección, se devuelve -1, lo que significa que la colección no se busca.
- Si *startFrom*  $< 0$ , se considera el desplazamiento desde el final de la colección (*startFrom:=startFrom+length*).  
Nota: incluso si *startFrom* es negativo, la colección se sigue buscando de izquierda a derecha.
- Si *startFrom* = 0, se busca en toda la colección (por defecto).

### Ejemplo

```
var $col : Collection
var $i : Integer
$col:=New collection(1;2;"Henry";5;3;"Albert";6;4;"Alan";5)
$i:=$col.indexOf(3) // $i=4
$i:=$col.indexOf(5;5) // $i=9
$i:=$col.indexOf("al@") // $i=5
$i:=$col.indexOf("Hello") // $i=-1
```

## .indices()

► Histórico

.indices( *queryString* : Text { ; ...*value* : any } ) : Collection

Parámetros	Tipo		Descripción
<i>queryString</i>	Texto	->	Criterio de búsqueda
<i>value</i>	any	->	Valor(es) a comparar cuando se utiliza(n) marcador(es) de posición
Resultado	Collection	<-	Índice(s) de elementos que coinciden con <i>queryString</i> en la colección

### Descripción

La función `.indices()` funciona exactamente igual que la función `.query()` pero devuelve los índices, en la colección

original, de los elementos de la colección de objetos que coinciden con las condiciones de búsqueda *queryString*, y no los elementos en sí. Los índices se devuelven en orden ascendente.

Esta función no modifica la colección original.

El parámetro *queryString* utiliza la siguiente sintaxis:

```
propertyPath comparator value {logicalOperator propertyPath comparator value}
```

Para una descripción detallada de los parámetros *queryString* y *value*, consulte la función `dataClass.query()`.

## Ejemplo

```
var $c; $icol : Collection
$c:=New collection
$c.push(New object("name";"Cleveland";"zc";35049))
$c.push(New object("name";"Blountsville";"zc";35031))

$c.push(New object("name";"Adger";"zc";35006))
$c.push(New object("name";"Clanton";"zc";35046))
$c.push(New object("name";"Clanton";"zc";35045))
$icol:=$c.indices("name = :1";"Cleveland") // $icol=[0]
$icol:=$c.indices("zc > 35040") // $icol=[0,3,4]
```

## .insert()

► Histórico

`.insert( index : Integer ; element : any ) : Collection`

Parámetros	Tipo		Descripción
index	Integer	->	Dónde insertar el elemento
element	any	->	Elemento a insertar en la colección
Resultado	Collection	<-	Colección original que contiene los elementos insertados

## Descripción

La función `.insert()` inserta *element* en la posición *index* especificada en la instancia de la colección y devuelve la colección editada.

Esta función modifica la colección original.

En *index*, pase la posición donde quiere que se inserte el elemento en la colección.

Atención: recuerde que los elementos de la colección están numerados desde 0.

- Si *índice* > la longitud de la colección, el índice inicial real se fijará en la longitud de la colección.
- Si *index* < 0, se recalcula como *index*:=*index*+*length* (se considera el desplazamiento desde el final de la colección).
- Si el valor calculado es negativo, *index* toma el valor 0.

Se puede insertar cualquier tipo de elemento aceptado por una colección, incluso otra colección.

## Ejemplo

```

var $col : Collection
$col:=New collection("a";"b";"c";"d") // $col=["a","b","c","d"]
$col.insert(2;"X") // $col=["a","b","X","c","d"]
$col.insert(-2;"Y") // $col=["a","b","X","Y","c","d"]
$col.insert(-10;"Hi") // $col=["Hi","a","b","X","Y","c","d"]

```

## .join()

► Histórico

`join( delimiter : Text { ; option : Integer } ) : Text`

Parámetros	Tipo		Descripción
delimiter	Texto	->	Separador a utilizar entre elementos
option	Integer	->	<code>ck ignore null or empty</code> : ignorar las cadenas nulas y vacías en el resultado
Resultado	Texto	<-	Cadena que contiene todos los elementos de la colección, separados por un delimitador

### Descripción

La función `.join()` convierte todos los elementos de la colección en cadenas y las concatena utilizando la cadena `delimiter` especificada como separador. La función devuelve la cadena resultante.

Esta función no modifica la colección original.

Por defecto, los elementos nulos o vacíos de la colección se devuelven en la cadena resultante. Pase la constante `ck ignore null or empty` en el parámetro `option` si quiere eliminarlos de la cadena resultante.

### Ejemplo

```

var $c : Collection
var $t1;$t2 : Text
$c:=New collection(1;2;3;"Paris";Null;"";4;5)
$t1:=$c.join("|") //1|2|3|Paris|null||4|5
$t2:=$c.join("|";ck ignore null or empty) //1|2|3|Paris|4|5

```

## .lastIndexOf()

► Histórico

`lastIndexOf( toSearch : expression { ; startFrom : Integer } ) : Integer`

Parámetros	Tipo		Descripción
toSearch	expresión	->	El elemento que se va a buscar dentro de la colección
startFrom	Integer	->	Índice para iniciar la búsqueda en
Resultado	Integer	<-	Índice de la última ocurrencia de <code>toSearch</code> en la colección, -1 si no se encuentra

### Descripción

La función `.lastIndexOf()` busca la expresión `toSearch` entre los elementos de la colección y devuelve el índice de la última ocurrencia, o -1 si no se encontró.

Esta función no modifica la colección original.

En `toSearch`, pase la expresión a encontrar en la colección. Puede pasar:

- un valor escalar (texto, número, booleano, fecha),
- el valor null,
- una referencia de objeto o de colección.

`toSearch` debe coincidir exactamente con el elemento a encontrar (se aplican las mismas reglas que para el operador de igualdad).

Opcionalmente, puede pasar el índice de la colección desde el cual iniciar una búsqueda en reversa en `startFrom`.

- Si `startFrom >=` la longitud de la colección menos uno (`coll.length-1`), se busca en toda la colección (por defecto).
- Si `startFrom < 0`, se recalcula como `startFrom:=startFrom+length` (se considera el desplazamiento desde el final de la colección). Si el valor calculado es negativo, se devuelve -1 (no se busca en la colección). Nota: incluso si `startFrom` es negativo, la colección se sigue buscando de derecha a izquierda.
- Si `startFrom = 0`, se devuelve -1 lo que significa que la colección no se busca.

## Ejemplo

```
var $col : Collection
var $pos1;$pos2;$pos3;$pos4;$pos5 : Integer
$col:=Split string("a,b,c,d,e,f,g,h,i,j,e,k,e;,,") // $col.length=13
$pos1:=$col.lastIndex0f("e") //devuelve 12
$pos2:=$col.lastIndex0f("e";6) //devuelve 4
$pos3:=$col.lastIndex0f("e";15) //devuelve 12
$pos4:=$col.lastIndex0f("e";-2) //devuelve 10
$pos5:=$col.lastIndex0f("x") //devuelve -1
```

## .length

► Histórico

`.length`: Integer

### Descripción

La propiedad `.length` devuelve el número de elementos en la colección.

La propiedad `.length` se inicializa cuando se crea la colección. Añadir o eliminar elementos actualiza la longitud, si es necesario. Esta propiedad es sólo lectura (no se puede utilizar para definir el tamaño de la colección).

## Ejemplo

```
var $col : Collection // $col.length inicializada en 0
$col:=New collection("one";"two";"three") // $col.length actualizada a 3
$col[4]:="five" // $col.length actualizada a 5
$vSize:=$col.remove(0;3).length // $vSize=2
```

## .map()

► Histórico

`.map( methodName : Text { ; ...param : any } ) : Collection`

Parámetros	Tipo		Descripción
methodName	Texto	->	Nombre del método utilizado para transformar los elementos de la colección
param	any	->	Parámetros del método
Resultado	Collection	<-	Colección de valores transformados

## Descripción

La función `.map()` crea una nueva colección basada en el resultado de la llamada al método *methodName* sobre cada elemento de la colección original. Opcionalmente, puede pasar parámetros a *methodName* utilizando el(los) parámetro(s) *param*. `.map()` siempre devuelve una colección con el mismo tamaño que la colección original.

Esta función no modifica la colección original.

En *methodName*, pase el nombre del método a utilizar para evaluar los elementos de la colección, junto con su(s) parámetro(s) en *param* (opcional). En *methodName*, pase el nombre del método a utilizar para evaluar los elementos de la colección, junto con su(s) parámetro(s) en *param* (opcional).

*methodName* recibe los siguientes parámetros:

- en `$1.value` (todo tipo): valor del elemento que se va a asignar
- en `$2` (todo tipo): *param*
- en `$N...` (any type): *paramN...*

*methodName* define el(los) siguiente(s) parámetro(s):

- `$1.result` (todo tipo): nuevo valor transformado para añadir a la colección resultante
- `$1.stop` (boolean): true para detener la retrollamada del método. El valor devuelto es el último calculado.

## Ejemplo

```
var $c; $c2 : Collection  
$c:=New collection(1;4;9;10;20)  
$c2:=$c.map("Percentage";$c.sum())  
//$c2=[2.27,9.09,20.45,22.73,45.45]
```

Aquí está el método *Percentage*:

```
var $1 : Object  
var $2 : Real  
$1.result:=Round(( $1.value/$2)*100;2)
```

## .max()

► Histórico

`.max( { propertyPath : Text } ) : any`

Parámetros	Tipo		Descripción
propertyPath	Texto	->	Ruta de la propiedad del objeto que se utilizará para la evaluación
Resultado	Boolean, Text, Number, Collection, Object, Date	<-	Valor máximo en la colección

## Descripción

La función `.max()` devuelve el elemento con el valor más alto de la colección (el último elemento de la colección tal y como se ordenaría de forma ascendente utilizando la función `.sort()` ).

Esta función no modifica la colección original.

Si la colección contiene diferentes tipos de valores, la función `.max()` devolverá el valor máximo dentro del último tipo de elemento en el orden de la lista de tipos (ver la descripción de `.sort()` ).

Si la colección contiene objetos, pase el parámetro *propertyPath* para indicar la propiedad del objeto cuyo valor máximo desea obtener.

Si la colección está vacía, `.max()` devuelve *Undefined*.

## Ejemplo

```
var $col : Collection
$col:=New collection(200;150;55)
$col.push(New object("name";"Smith";"salary";10000))
$col.push(New object("name";"Wesson";"salary";50000))
$col.push(New object("name";"Alabama";"salary";10500))
$max:=$col.max() //{name:Alabama,salary:10500}
$maxSal:=$col.max("salary") //50000
$maxName:=$col.max("name") //"Wesson"
```

## .min()

► Histórico

`.min( { propertyPath : Text } ) : any`

Parámetros	Tipo		Descripción
propertyPath	Texto	->	Ruta de la propiedad del objeto que se utilizará para la evaluación
Resultado	Boolean, Text, Number, Collection, Object, Date	<-	Valor mínimo en la colección

### Descripción

La función `.min()` devuelve el elemento con el valor más pequeño de la colección (el primer elemento de la colección tal y como se ordenaría de forma ascendente utilizando la función `.sort()` ).

Esta función no modifica la colección original.

Si la colección contiene diferentes tipos de valores, la función `.min()` devolverá el valor mínimo dentro del primer tipo de elemento en el orden de la lista de tipos (ver la descripción de `.sort()` ).

Si la colección contiene objetos, pase el parámetro *propertyPath* para indicar la propiedad del objeto cuyo valor mínimo desea obtener.

Si la colección está vacía, `.min()` devuelve *Undefined*.

## Ejemplo

```
var $col : Collection
$col:=New collection(200;150;55)
$col.push(New object("name";"Smith";"salary";10000))
$col.push(New object("name";"Wesson";"salary";50000))
$col.push(New object("name";"Alabama";"salary";10500))
$min:=$col.min() //55
$minSal:=$col.min("salary") //10000
$minName:=$col.min("name") //"Alabama"
```

## .orderBy()

► Histórico

```
.orderBy( ) : Collection
.orderBy( pathStrings : Text ) : Collection
.orderBy( pathObjects : Collection ) : Collection
.orderBy( ascOrDesc : Integer ) : Collection
```

Parámetros	Tipo		Descripción
pathStrings	Texto	->	Ruta(s) de propiedad(es) a utilizar para ordenar la colección
pathObjects	Collection	->	Colección de objetos criterio
ascOrDesc	Integer	->	<code>ck ascending</code> o <code>ck descending</code> (valores escalares)
Resultado	Collection	<-	Copia ordenada de la colección (copia superficial)

## Descripción

La función `.orderBy()` devuelve una nueva colección que contiene todos los elementos de la colección en el orden especificado.

Esta función devuelve una *copia superficial*, lo que significa que los objetos o colecciones de ambas colecciones comparten la misma referencia. Si la colección original es una colección compartida, la colección devuelta es también una colección compartida.

Esta función no modifica la colección original.

Si no se pasa ningún parámetro, la función ordena los valores escalares de la colección en orden ascendente (otros tipos de elementos, como objetos o colecciones, se devuelven desordenados). Puede modificar este orden automático pasando las constantes `ck ascending` o `ck descending` en el parámetro `ascOrDesc` (ver más abajo).

También puede pasar un parámetro de criterios para definir cómo deben ordenarse los elementos de la colección. Se admiten tres sintaxis para este parámetro:

- `pathStrings`: Texto (fórmula). Sintaxis: `propertyPath1 {desc or asc}, propertyPath2 {desc or asc},...` (orden por defecto: `asc`). `pathStrings` contiene una fórmula compuesta de 1 a x rutas de propiedades y (opcionalmente) órdenes de clasificación, separados por comas. El orden en que se pasan las propiedades determina la prioridad de ordenación de los elementos de la colección. Por defecto, las propiedades se clasifican en orden ascendente. Puede definir el orden de clasificación de una propiedad en la cadena de criterios, separado de la ruta de la propiedad por un solo espacio: pase "`asc`" para ordenar en orden ascendente o "`desc`" en orden descendente.
- `pathObjects` : Collection. Puede añadir tantos objetos en la colección `pathObjects` como sea necesario. Por defecto, las propiedades se clasifican en orden ascendente ("`descending`" es `false`). Cada elemento de la colección contiene un objeto estructurado de la siguiente manera:

```
{
  "propertyPath": string,
  "descending": boolean
}
```

- `ascOrDesc`: Integer. Se pasa una de las siguientes constantes del tema `Objects and collections`:

Constante	Tipo	Valor	Comentario
<code>ck ascending</code>	Entero largo	0	Los elementos se ordenan de forma ascendente (por defecto)
<code>ck descending</code>	Entero largo	1	Los elementos se ordenan de forma descendente

Esta sintaxis sólo ordena los valores escalares de la colección (otros tipos de elementos, como objetos o colecciones, se devuelven desordenados).

Si la colección contiene elementos de diferentes tipos, se agrupan primero por tipo y se ordenan después. Si `attributePath` lleva a una propiedad de objeto que contiene valores de diferentes tipos, primero se agrupan por tipo y se ordenan después.

1. null
2. booleanos
3. strings
4. numbers
5. objetos
6. collections
7. fechas

### Ejemplo 1

Ordenar una colección de números de forma ascendente y descendente:

```
var $c; $c2; $3 : Collection
$c:=New collection
For($vCounter;1;10)
    $c.push(Random)
End for
$c2:=$c.orderBy(ck ascending)
$c3:=$c.orderBy(ck descending)
```

### Ejemplo 2

Ordenar una colección de objetos a partir de una fórmula de texto con nombres de propiedades:

```
var $c; $c2 : Collection
$c:=New collection
For($vCounter;1;10)
    $c.push(New object("id";$vCounter;"value";Random))
End for
$c2:=$c.orderBy("value desc")
$c2:=$c.orderBy("value desc, id")
$c2:=$c.orderBy("value desc, id asc")
```

Ordenar una colección de objetos con una ruta de propiedades:

```
var $c; $c2 : Collection
$c:=New collection
$c.push(New object("name";"Cleveland";"phones";New object("p1";"01";"p2";"02")))
$c.push(New object("name";"Blountsville";"phones";New object("p1";"00";"p2";"03")))

$c2:=$c.orderBy("phones.p1 asc")
```

### Ejemplo 3

Ordenar una colección de objetos utilizando una colección de objetos criterio:

```
var $crit; $c; $c2 : Collection
$crit:=New collection
$c:=New collection
For($vCounter;1;10)
    $c.push(New object("id";$vCounter;"value";Random))
End for
$crit.push(New object("propertyPath";"value";"descending";True))
$crit.push(New object("propertyPath";"id";"descending";False))
$c2:=$c.orderBy($crit)
```

Ordenar con una ruta de propiedad:

```

var $crit; $c; $c2 : Collection
$c:=New collection
$c.push(New object("name";"Cleveland";"phones";New object("p1";"01";"p2";"02")))
$c.push(New object("name";"Blountsville";"phones";New object("p1";"00";"p2";"03")))
$crit:=New collection(New object("propertyPath";"phones.p2";"descending";True))
$c2:=$c.orderBy($crit)

```

## .orderByMethod()

► Histórico

.orderByMethod( *methodName* : Text { ; ...*extraParam* : expression } ) : Collection

Parámetros	Tipo		Descripción
<i>methodName</i>	Texto	->	Nombre del método utilizado para especificar el orden de clasificación
<i>extraParam</i>	expresión	->	Parámetros del método
Resultado	Collection	<-	Copia ordenada de la colección (copia superficial)

### Descripción

La función `.orderByMethod()` devuelve una nueva colección que contiene todos los elementos de la colección en el orden definido a través del método *methodName*.

Esta función devuelve una *copia superficial*, lo que significa que los objetos o colecciones de ambas colecciones comparten la misma referencia. Si la colección original es una colección compartida, la colección devuelta es también una colección compartida.

Esta función no modifica la colección original.

En *methodName*, pase un método de comparación que compare dos valores y devuelva `true` en `$1.result` si el primer valor es menor que el segundo. Puede suministrar parámetros adicionales a *methodName* si es necesario.

- *methodName* recibirá los siguientes parámetros:
  - `$1` (objeto), donde:
    - `$1.value` (todo tipo): primer valor del elemento a comparar
    - `$1.value2` (todo tipo): segundo valor del elemento a comparar
  - `$2...$N` (cualquier tipo): parámetros adicionales
- *methodName* define el siguiente parámetro:
  - `$1.result` (boolean): `true` si `$1.value < $1.value2`, `false` de lo contrario

### Ejemplo 1

Desea ordenar una colección de cadenas en orden numérico en lugar de orden alfabético:

```

var $c; $c2; $c3 : Collection
$c:=New collection
$c.push("33";"4";"1111";"222")
$c2:=$c.orderBy() // $c2=[1111,"222","33","4"], alphabetical order
$c3:=$c.orderByMethod("NumAscending") // $c3=[4,"33","222","1111"]

```

El código para *NumAscending* es:

```

$c1.result:=Num($1.value)<Num($1.value2)

```

### Ejemplo 2

Quiere ordenar una colección de cadenas según su longitud:

```
var $fruits; $c2 : Collection  
$fruits:=New collection("Orange";"Apple";"Grape";"pear";"Banana";"fig";"Blackberry";"Passion fruit")  
$c2:=$fruits.orderByMethod("WordLength")  
//c2=[Passion fruit,Blackberry,Orange,Banana,Apple,Grape,pear,fig]
```

El código para *WordLength* es:

```
$1.result:=Length(String($1.value))>Length(String($1.value2))
```

### Ejemplo 3

Ordenar los elementos de la colección por código de caracteres o alfabéticamente:

```
var $strings1; $strings2 : Collection  
$strings1:=New collection("Alpha";"Charlie";"alpha";"bravo";"Bravo";"charlie")  
  
//utilizando el código de caracteres:  
$strings2:=$strings1.orderByMethod("sortCollection";sk character codes)  
// resultado : ["Alpha","Bravo","Charlie","alpha","bravo","charlie"]  
  
//utilizando el lenguaje:  
$strings2:=$strings1.orderByMethod("sortCollection";sk strict)  
// resultado : ["alpha","Alpha","bravo","Bravo","charlie","Charlie"]
```

El método *sortCollection*:

```
var$1Object  
var$2Integer // opción de ordenación  
  
$1.result:=(Compare strings($1.value;$1.value2;$2)<0)
```

## .pop()

► Histórico

.pop() : any

Parámetros	Tipo		Descripción
Resultado	any	<-	Último elemento de la colección

### Descripción

La función `.pop()` elimina el último elemento de la colección y lo devuelve como resultado de la función.

Esta función modifica la colección original.

Cuando se aplica a una colección vacía, `.pop()` devuelve *undefined*.

### Ejemplo

`.pop()`, utilizado junto con `.push()`, puede utilizarse para implementar una funcionalidad primera entrada, última salida de tratamiento de datos apilados:

```

var $stack : Collection
$stack:=New collection //$/stack=[]
$stack.push(1;2) //$/stack=[1,2]
$stack.pop() //$/stack=[1] Devuelve 2
$stack.push(New collection(4;5)) //$/stack=[[1,[4,5]]
$stack.pop() //$/stack=[1] Devuelve [4,5]
$stack.pop() //$/stack=[] Devuelve 1

```

## .push()

► Histórico

.push( *element* : any { ;...*elementN* } ) : Collection

Parámetros	Tipo		Descripción
element	Mixed	->	Elemento(s) a añadir a la colección
Resultado	Collection	<-	Colección original que contiene los elementos añadidos

### Descripción

La función `.push()` añade uno o más *elementos* al final de la instancia de la colección y devuelve la colección editada.

Esta función modifica la colección original.

### Ejemplo 1

```

var $col : Collection
$col:=New collection(1;2) //$/col=[1,2]
$col.push(3) //$/col=[1,2,3]
$col.push(6;New object("firstname";"John";"lastname";"Smith"))
//$/col=[1,2,3,6,{firstname:John,lastname:Smith}]

```

### Ejemplo 2

Quiere ordenar la colección resultante:

```

var $col; $sortedCol : Collection
$col:=New collection(5;3;9) //$/col=[5,3,9]
$sortedCol:=$col.push(7;50).sort()
//$/col=[5,3,9,7,50]
//$/sortedCol=[3,5,7,9,50]

```

## .query()

► Histórico

.query( *queryString* : Text ; ...*value* : any ) : Collection

.query( *queryString* : Text ; *querySettings* : Object ) : Collection

Parámetros	Tipo		Descripción
queryString	Texto	->	Criterio de búsqueda
value	Mixed	->	Valor(es) a comparar cuando se utiliza(n) marcador(es) de posición
querySettings	Objeto	->	Opciones de búsqueda: parámetros, atributos
Resultado	Collection	<-	Elemento(s) que coincide(n) con queryString en la colección

## Descripción

La función `.query()` devuelve todos los elementos de una colección de objetos que coinciden con las condiciones de búsqueda definidas por `queryString` y (opcionalmente) `value` o `querySettings`. Si la colección original es una colección compartida, la colección devuelta es también una colección compartida.

Esta función no modifica la colección original.

El parámetro `queryString` utiliza la siguiente sintaxis:

```
valor del comparador propertyPath {logicalOperator propertyPath comparator value}
```

Para obtener información detallada sobre cómo construir una consulta utilizando los parámetros `queryString`, `value` y `querySettings`, consulte la descripción de la función `dataClass.query\(\)`.

Las fórmulas no son soportadas por la función `collection.query()`, ni en el parámetro `queryString` ni como parámetro del objeto `formula`.

## Ejemplo 1

```
var $c; $c2; $c3 : Collection
$c:=New collection
$c.push(New object("name";"Cleveland";"zc";35049))
$c.push(New object("name";"Blountsville";"zc";35031))
$c.push(New object("name";"Adger";"zc";35006))
$c.push(New object("name";"Clanton";"zc";35046))
$c.push(New object("name";"Clanton";"zc";35045))
$c2:=$c.query("name = :1";"Cleveland") // $c2=[{name:Cleveland,zc:35049}]
$c3:=$c.query("zc > 35040") // $c3=[{name:Cleveland,zc:35049},{name:Clanton,zc:35046},{name:Clanton,zc:35045}]
```

## Ejemplo 2

```
var $c : Collection
$c:=New collection
$c.push(New object("name";"Smith";"dateHired";!22-05-2002!;"age";45))
$c.push(New object("name";"Wesson";"dateHired";!30-11-2017!))
$c.push(New object("name";"Winch";"dateHired";!16-05-2018!;"age";36))

$c.push(New object("name";"Sterling";"dateHired";!10-5-1999!;"age";Null))
$c.push(New object("name";"Mark";"dateHired";!01-01-2002!))
```

Este ejemplo devuelve las personas cuyo nombre contiene "in":

```
$col:=$c.query("name = :1";"@in@")
// $col=[{name:Winch...},{name:Sterling...}]
```

Este ejemplo devuelve las personas cuyo nombre no empieza por una cadena de una variable (introducida por el usuario, por ejemplo):

```
$col:=$c.query("name # :1;$aString+"@")
//if $astring="W"
//$col=[{name:Smith...},{name:Sterling...},{name:Mark...}]
```

Este ejemplo devuelve las personas cuya edad no se conoce (propiedad definida como null o indefinida):

```
$col:=$c.query("age=null") //no están permitidos los marcadores de posición con "null"
//$col=[{name:Wesson...},{name:Sterling...},{name:Mark...}]
```

Este ejemplo devuelve las personas contratadas hace más de 90 días:

```
$col:=$c.query("dateHired < :1";(Current date-90))
//$col=[{name:Smith...},{name:Sterling...},{name:Mark...}] si hoy es 01/10/2018
```

### Ejemplo 3

Se pueden encontrar más ejemplos de búsquedas en la página `dataClass.query()`.

## .reduce()

► Histórico

`.reduce( methodName : Text ) : any`

`.reduce( methodName : Text ; initialValue : any { ; ...param : expression } ) : any`

Parámetros	Tipo		Descripción
methodName	Texto	->	Nombre de la función a la que se llama para procesar los elementos de la colección
initialValue	Text, Number, Object, Collection, Date, Boolean	->	Valor a utilizar como primer argumento de la primera llamada de <i>methodName</i>
param	expresión	->	Parámetro(s) a pasar a <i>methodName</i>
Resultado	Text, Number, Object, Collection, Date, Boolean	<-	Resultado del valor del acumulador

### Descripción

La función `.reduce()` aplica el método de retrollamada *methodName* contra un acumulador y cada elemento de la colección (de izquierda a derecha) para reducirlo a un único valor.

Esta función no modifica la colección original.

En *methodName*, pase el nombre del método a utilizar para evaluar los elementos de la colección, junto con su(s) parámetro(s) en *param* (opcional). *methodName* toma cada elemento de la colección y realiza todas las operaciones deseadas para acumular el resultado en `$1.accumulator`, que se devuelve en `$1.value`.

Puede pasar el valor para inicializar el acumulador en *initialValue*. Si se omite, `$1.accumulator` comienza con *Undefined*.

*methodName* recibe los siguientes parámetros:

- en `$1.value`: valor del elemento a procesar
- en `$2: param`
- en `$N...: paramN...`

*methodName* define el(los) siguiente(s) parámetro(s):

- *\$1.accumulator*: valor que va a ser modificado por la función y que es inicializado por *initValue*.
- *\$1.stop* (boolean, opcional): true para detener la retrollamada del método. El valor devuelto es el último calculado.

## Ejemplo 1

```
C_COLLECTION($c)
$c:=New collection(5;3;5;1;3;4;4;6;2;2)
$r:=$c.reduce("Multiply";1) //devuelve 86400
```

Con el siguiente método *Multiply*:

```
If(Value type($1.value)=Is real)
    $1.accumulator:=$1.accumulator*$1.value
End if
```

## Ejemplo 2

Este ejemplo permite reducir varios elementos de la colección a uno solo:

```
var $c;$r : Collection
$c:=New collection
$c.push(New collection(0;1))
$c.push(New collection(2;3))
$c.push(New collection(4;5))
$c.push(New collection(6;7))
$r:=$c.reduce("Flatten") //r=[0,1,2,3,4,5,6,7]
```

Con el siguiente método *Flatten*:

```
If($1.accumulator=Null)
    $1.accumulator:=New collection
End if
$1.accumulator.combine($1.value)
```

## .remove()

► Histórico

.remove( *index* : Integer { ; *howMany* : Integer } ) : Collection

Parámetros	Tipo		Descripción
<i>index</i>	Integer	->	Elemento en el que se inicia la eliminación
<i>howMany</i>	Integer	->	Número de elementos a eliminar, o 1 elemento si se omite
Resultado	Collection	<-	Colección original sin elementos eliminados

### Descripción

La función `.remove()` elimina uno o más elementos de la posición *index* especificada en la colección y devuelve la colección editada.

Esta función modifica la colección original.

En *index*, pase la posición donde quiere eliminar el elemento de la colección.

Atención: recuerde que los elementos de la colección están numerados desde 0. Si *índice* es mayor que la longitud de la colección, el índice inicial real se fijará en la longitud de la colección.

- Si *index* < 0, se recalcula como *index*:=*index*+*length* (se considera el desplazamiento desde el final de la colección).
- Si el valor calculado < 0, *index* toma el valor 0.
- Si el valor calculado > la longitud de la colección, *index* se establece para la longitud.

En *howMany*, pase el número de elementos a eliminar de *index*. Si no se especifica *howMany*, se elimina un elemento.

Si se intenta eliminar un elemento de una colección vacía, el método no hace nada (no se genera ningún error).

## Ejemplo

```
var $col : Collection  
$col:=New collection("a";"b";"c";"d";"e";"f";"g";"h")  
$col.remove(3) // $col=["a","b","c","e","f","g","h"]  
$col.remove(3;2) // $col=["a","b","c","g","h"]  
$col.remove(-8;1) // $col=["b","c","g","h"]  
$col.remove(-3;1) // $col=["b","g","h"]
```

## .resize()

► Histórico

.resize( *size* : Integer { ; *defaultValue* : any } ) : Collection

Parámetros	Tipo		Descripción
<i>size</i>	Integer	->	Nuevo tamaño de la colección
<i>defaultValue</i>	Number, Text, Object, Collection, Date, Boolean	->	Valor por defecto para llenar nuevos elementos
Resultado	Collection	<-	Colección original redimensionada

### Descripción

La función `.resize()` ajusta la longitud de la colección al nuevo tamaño especificado y devuelve la colección redimensionada.

Esta función modifica la colección original.

- Si *size* < la longitud de la colección, los elementos que exceden se eliminan de la colección.
- Si *size* > longitud de la colección, la longitud de la colección se incrementa al tamaño.

Por defecto, los nuevos elementos se llenan con valores null. Puede especificar el valor a llenar en los elementos añadidos utilizando el parámetro *defaultValue*.

## Ejemplo

```

var $c : Collection
$c:=New collection
$c.resize(10) // $c=[null,null,null,null,null,null,null,null,null,null]

$c:=New collection
$c.resize(10;0) // $c=[0,0,0,0,0,0,0,0,0,0]

$c:=New collection(1;2;3;4;5)
$c.resize(10;New object("name";"X")) // $c=[1,2,3,4,5,{name:X},{name:X},{name:X},{name:X}]

$c:=New collection(1;2;3;4;5)
$c.resize(2) // $c=[1,2]

```

## .reverse()

► Histórico

.reverse() : Collection

Parámetros	Tipo		Descripción
Resultado	Collection	<-	Copia invertida de la colección

### Descripción

La función `.reverse()` devuelve una copia profunda de la colección con todos sus elementos en orden inverso. Si la colección original es una colección compartida, la colección devuelta es también una colección compartida.

Esta función no modifica la colección original.

### Ejemplo

```

var $c; $c2 : Collection
$c:=New collection(1;3;5;2;4;6)
$c2:=$c.reverse() // $c2=[6,4,2,5,3,1]

```

## .shift()

► Histórico

.shift() : any

Parámetros	Tipo		Descripción
Resultado	any	<-	Primer elemento de la colección

### Descripción

La función `.shift()` elimina el primer elemento de la colección y lo devuelve como resultado de la función.

Esta función modifica la colección original.

Si la colección está vacía, este método no hace nada.

### Ejemplo

```

var $c : Collection
var $val : Variant
$c:=New collection(1;2;4;5;6;7;8)
$val:=$c.shift()
// $val=1
// $c=[2,4,5,6,7,8]

```

## .slice()

► Histórico

.slice( *startFrom* : Integer { ; *end* : Integer } ) : Collection

Parámetros	Tipo		Descripción
<i>startFrom</i>	Integer	->	Índice de inicio (incluido)
<i>end</i>	Integer	->	Índice final (no incluido)
Resultado	Collection	<-	Nueva colección que contiene elementos cortados (copia superficial)

### Descripción

La función `.slice()` devuelve una porción de una colección en una nueva colección, seleccionada desde el índice *startFrom* hasta el índice *end* (no incluye el final). Esta función devuelve una *copia superficial* de la colección. Si la colección original es una colección compartida, la colección devuelta es también una colección compartida.

Esta función no modifica la colección original.

La colección devuelta contiene el elemento especificado por *startFrom* y todos los elementos subsiguientes hasta, pero sin incluir, el elemento especificado por *end*. Si sólo se especifica el parámetro *startFrom*, la colección devuelta contiene todos los elementos desde *startFrom* hasta el último elemento de la colección original.

- Si *startFrom* < 0, se recalcula como *startFrom*:=*startFrom*+*length* (se considera el desplazamiento desde el final de la colección).
- Si el valor calculado < 0, *startFrom* toma el valor 0.
- Si *end* < 0 , se recalcula como *end*:=*end*+*length*.
- Si *end* < *startFrom* (valores pasados o calculados), el método no hace nada.

### Ejemplo

```

var $c; $nc : Collection
$c:=New collection(1;2;3;4;5)
$nc:=$c.slice(0;3) //nc=[1,2,3]
$nc:=$c.slice(3) //nc=[4,5]
$nc:=$c.slice(1;-1) //nc=[2,3,4]
$nc:=$c.slice(-3;-2) //nc=[3]

```

## .some()

► Histórico

.some( *methodName* : Text { ; ...*param* : any } ) : Boolean

.some( *startFrom* : Integer ; *methodName* : Text { ; ...*param* : any } ) : Boolean

Parámetros	Tipo		Descripción
startFrom	Integer	->	Índice para iniciar la prueba en
methodName	Texto	->	Nombre del método a llamar para la prueba
param	Mixed	->	Parámetro(s) a pasar a <i>methodName</i>
Resultado	Booleano	<-	True si al menos un elemento ha superado la prueba con éxito

## Descripción

La función `.some()` devuelve true si al menos un elemento de la colección ha superado con éxito una prueba implementada en el método *methodName* proporcionado.

En *methodName*, pase el nombre del método a utilizar para evaluar los elementos de la colección, junto con su(s) parámetro(s) en *param* (opcional). *methodName* puede realizar cualquier prueba, con o sin los parámetros. En *methodName*, pase el nombre del método a utilizar para evaluar los elementos de la colección, junto con su(s) parámetro(s) en *param* (opcional).

*methodName* recibe los siguientes parámetros:

- en `$1.value`: valor del elemento a evaluar
- en `$2`: param
- en `$N...: param2...paramN`

*methodName* define el(los) siguiente(s) parámetro(s):

- `$1.result` (boolean): true si la evaluación del valor del elemento tiene éxito, si no false.
- `$1.stop` (boolean, opcional): true para detener la retrollamada del método. El valor devuelto es el último calculado.

En algún caso, en el momento en que la función `.some()` encuentre el primer elemento de la colección que devuelva true en `$1.result`, deja de llamar a *methodName* y devuelve false.

Por defecto, `.some()` comprueba toda la colección. Opcionalmente, puede pasar el índice de un elemento desde el cual iniciar la prueba en *startFrom*.

- Si *startFrom*  $\geq$  la longitud de la colección, se devuelve False, lo que significa que la colección no se prueba.
- Si *startFrom*  $< 0$ , se considera como el desplazamiento desde el final de la colección.
- Si *startFrom* = 0, se busca en toda la colección (por defecto).

## Ejemplo

```
var $c : Collection
var $b : Boolean
$c:=New collection
$c.push(-5;-3;-1;-4;-6;-2)
$b:=$c.some("NumberGreaterThan0") // devuelve false
$c.push(1)
$b:=$c.some("NumberGreaterThan0") // devuelve true

$c:=New collection
$c.push(1;-5;-3;-1;-4;-6;-2)
$b:=$c.some("NumberGreaterThan0") // $b=true
$b:=$c.some(1;"NumberGreaterThan0") // $b=false
```

Con el siguiente método *NumberGreaterThan0*:

```
$1.result:=$1.value>0
```

## `.sort()`

► Histórico

.sort( *methodName* : Text { ; ...*extraParam* : any } ) : Collection

Parámetros	Tipo		Descripción
methodName	Texto	->	Nombre del método utilizado para especificar el orden de clasificación
extraParam	any	->	Parámetros del método
Resultado	Collection	<-	Colección original ordenada

## Descripción

La función `.sort()` ordena los elementos de la colección original y también devuelve la colección ordenada.

Esta función modifica la colección original.

Si se llama a `.sort()` sin parámetros, sólo se ordenan los valores escalares (número, texto, fecha, booleanos). Los elementos se ordenan por defecto de forma ascendente, según su tipo.

Si desea ordenar los elementos de la colección en otro orden o clasificar cualquier tipo de elemento, debe suministrar en *methodName*, un método de comparación que compare dos valores y devuelva `true` en `$1.result` si el primer valor es menor que el segundo. Puede suministrar parámetros adicionales a *methodName* si es necesario.

- *methodName* recibirá los siguientes parámetros:
  - \$1 (objeto), donde:
    - `$1.value` (todo tipo): primer valor del elemento a comparar
    - `$1.value2` (todo tipo): segundo valor del elemento a comparar
  - \$2...\$N (cualquier tipo): parámetros adicionales

*methodName* define el siguiente parámetro: \* `$1.result` (boolean): true si `$1.value < $1.value2`, false de lo contrario

Si la colección contiene elementos de diferentes tipos, se agrupan primero por tipo y se ordenan después. Si *attributePath* lleva a una propiedad de objeto que contiene valores de diferentes tipos, primero se agrupan por tipo y se ordenan después.

1. null
2. booleanos
3. strings
4. numbers
5. objetos
6. collections
7. fechas

## Ejemplo 1

```
var $col; $col2 : Collection
$col:=New collection("Tom";5;"Mary";3;"Henry";1;"Jane";4;"Artie";6;"Chip";2)
$col2:=$col.sort() // $col2=[ "Artie","Chip","Henry","Jane","Mary","Tom",1,2,3,4,5,6]
// $col=[ "Artie","Chip","Henry","Jane","Mary","Tom",1,2,3,4,5,6]
```

## Ejemplo 2

```
var $col; $col2 : Collection
$col:=New collection(10;20)
$col2:=$col.push(5;3;1;4;6;2).sort() // $col2=[1,2,3,4,5,6,10,20]
```

## Ejemplo 3

```

var $col; $col2; $col3 : Collection
$col:=New collection(33;4;66;1111;222)
$col2:=$col.sort() //ordenación numérica: [4,33,66,222,1111]
$col3:=$col.sort("numberOrder") //ordenación alfabética: [1111,222,33,4,66]

```

```

//método proyecto numberOrder
var $1 : Object
$1.result:=String($1.value)<String($1.value2)

```

## .sum()

► Histórico

.sum( { *propertyPath* : Text } ) : Real

Parámetros	Tipo		Descripción
propertyPath	Texto	->	Ruta de la propiedad del objeto que se utilizará para el cálculo
Resultado	Real	<-	Suma de los valores de la colección

### Descripción

La función `.sum()` devuelve la suma de todos los valores de la instancia de la colección.

Para el cálculo sólo se tienen en cuenta los elementos numéricos (se ignoran otros tipos de elementos).

Si la colección contiene objetos, pasa el parámetro *propertyPath* para indicar la propiedad del objeto a tener en cuenta.

`.sum()` devuelve 0 si:

- la colección está vacía,
- la colección no contiene elementos numéricos,
- *propertyPath* no se encuentra en la colección.

### Ejemplo 1

```

var $col : Collection
var $vSum : Real
$col:=New collection(10;20;"Monday";True;2)
$vSum:=$col.sum() //32

```

### Ejemplo 2

```

var $col : Collection
var $vSum : Real
$col:=New collection
$col.push(New object("name";"Smith";"salary";10000))
$col.push(New object("name";"Wesson";"salary";50000))
$col.push(New object("name";"Gross";"salary";10500,5))
$vSum:=$col.sum("salary") //vSum=70500,5

```

## .unshift()

► Histórico

.unshift( *value* : any { ;...*valueN* : any } ) : Collection

Parámetros	Tipo		Descripción
value	Text, Number, Object, Collection, Date	->	Valor(es) a insertar al principio de la colección
Resultado	Real	<-	Colección que contiene los elementos añadidos

## Descripción

La función `.unshift()` inserta el(los) *valor(es)* dado(s) al principio de la colección y devuelve la colección modificada.

Esta función modifica la colección original.

Si se pasan varios valores, se insertan todos a la vez, lo que significa que aparecen en la colección resultante en el mismo orden que en la lista de argumentos.

## Ejemplo

```
var $c : Collection
$c:=New collection(1;2)
$c.unshift(4) // $c=[4,1,2]
$c.unshift(5) // $c=[5,4,1,2]
$c.unshift(6;7) // $c=[6,7,5,4,1,2]
```

# CryptoKey

La clase `CryptoKey` del lenguaje 4D contiene un par de llaves de cifrado asimétrico.

Esta clase está disponible en el almacén de clases de `4D`.

## Ejemplo

El siguiente código de ejemplo firma y verifica un mensaje utilizando un nuevo par de llaves ECDSA, por ejemplo para hacer un token web JSON ES256.

```
// Generar un nuevo par de llaves ECDSA
$key:=4D.CryptoKey.new(New object("type";"ECDSA";"curve";"prime256v1"))

// Obtener la firma como base64
$message:="hello world"
$signature:=$key.sign($message;New object("hash";"SHA256"))

// Verificar firma
$status:=$key.verify($message;$signature;New object("hash";"SHA256"))
ASSERT($status.success)
```

## Resumen

<code>4D.CryptoKey.new( settings : Object ) : 4D.CryptoKey</code>	crea un nuevo objeto <code>4D.CryptoKey</code> que encapsula un par de llaves de cifrado
<code>.curve : Text</code>	nombre de la curva normalizada de la llave
<code>.decrypt( message : Text ; options : Object ) : Object</code>	descifra el parámetro <i>message</i> utilizando la llave private
<code>.encrypt( message : Text ; options : Object ) : Text</code>	encripta el parámetro <i>message</i> utilizando la llave pública
<code>.getPrivateKey() : Text</code>	devuelve la llave privada del objeto <code>CryptoKey</code>
<code>.getPublicKey( ) : Text</code>	devuelve la llave pública del objeto <code>CryptoKey</code>
<code>.sign (message : Text ; options : Text) : Text</code>	firma la representación utf8 de un <i>message</i> string
<code>.size : Integer</code>	el tamaño de la llave en bits
<code>.type : Text</code>	Nombre del tipo de llave - "RSA", "ECDSA", "PEM"
<code>.verify( message : Text ; signature : Text ; options : Object) : object</code>	verifica la firma base64 contra la representación utf8 del <i>message</i>

## 4D.CryptoKey.new()

► Histórico

4D.CryptoKey.new( *settings* : Object ) : 4D.CryptoKey

Parámetros	Tipo		Descripción
parámetros	Objeto	->	Parámetros para generar o cargar un par de llaves
result	4D.CryptoKey	<-	Objeto que contiene un par de llaves de encriptación

La función `4D.CryptoKey.new()` crea un nuevo objeto `4D.CryptoKey` que encapsula un par de llaves de cifrado, basado en el parámetro del objeto *settings*. Permite generar una nueva llave RSA o ECDSA, o cargar un par de llaves existente desde una definición PEM.

### *parámetros*

Propiedad	Tipo	Descripción
<code>curve</code>	texto	Nombre de la curva ECDSA
<code>pem</code>	texto	Definición PEM de una llave de cifrado a cargar
<code>size</code>	integer	Tamaño de la llave RSA en bits
<code>type</code>	texto	Tipo de la llave: "RSA", "ECDSA" o "PEM"

### *CryptoKey*

El objeto `CryptoKey` devuelto encapsula un par de llaves de cifrado. Es un objeto compartido y, por tanto, puede ser utilizado por varios procesos 4D simultáneamente.

## .curve

► Histórico

.curve : Text

Definido sólo para las llaves ECDSA: el nombre de la curva normalizada de la llave. Generalmente "prime256v1" para ES256 (por defecto), "secp384r1" para ES384, "secp521r1" para ES512.

## .decrypt()

► Histórico

.decrypt( *message* : Text ; *options* : Object ) : Object

Parámetros	Tipo		Descripción
<code>message</code>	Texto	->	Cadena de mensajes a decodificar utilizando <code>options.encodingEncrypted</code> y descifrar.
<code>options</code>	Objeto	->	Opciones de decodificación
Resultado	Objeto	<-	Estado

La función `.decrypt()` descifra el parámetro *message* utilizando la llave private. El algoritmo utilizado depende del tipo de la llave.

La llave debe ser una llave RSA, el algoritmo es RSA-OAEP (ver [RFC 3447](#)).

### *options*

Propiedad	Tipo	Descripción
hash	texto	Algoritmo Digest a utilizar. Por ejemplo: "SHA256", "SHA384" o "SHA512".
encodingEncrypted	texto	Codificación utilizada para convertir el parámetro <code>mensaje</code> en la representación binaria a descifrar. Puede ser "Base64", o "Base64URL". Por defecto es "Base64".
encodingDecrypted	texto	Codificación utilizada para convertir el mensaje binario descifrado en la cadena de resultados. Puede ser "UTF-8", "Base64" o "Base64URL". Por defecto es "UTF-8".

## Resultado

La función devuelve un objeto "status" con la propiedad `success` definida como `true` si el `message` pudo ser descifrado con éxito.

Propiedad	Tipo	Descripción
<code>success</code>	booleano	True si el mensaje ha sido descifrado con éxito
<code>result</code>	texto	Mensaje descifrado y decodificado utilizando <code>options.encodingDecrypted</code>
<code>errors</code>	colección	Si <code>success</code> es <code>false</code> , puede contener una colección de errores

En caso de que el `message` no haya podido ser descifrado por no haber sido encriptado con la misma llave o algoritmo, el objeto `status` que se devuelve contiene una colección de errores en `status.errors`.

## .encrypt()

► Histórico

`.encrypt( message : Text ; options : Object ) : Text`

Parámetros	Tipo		Descripción
<code>message</code>	Texto	->	Cadena de mensajes a codificar utilizando <code>options.encodingDecrypted</code> y encriptado.
<code>options</code>	Objeto	->	Opciones de codificación
Resultado	Texto	<-	Mensaje encriptado y codificado utilizando la opción <code>options.encodingEncrypted</code>

La función `.encrypt()` encripta el parámetro `message` utilizando la llave pública. El algoritmo utilizado depende del tipo de la llave.

La llave debe ser una llave RSA, el algoritmo es RSA-OAEP (ver [RFC 3447](#)).

### options

Propiedad	Tipo	Descripción
<code>hash</code>	texto	Algoritmo Digest a utilizar. Por ejemplo: "SHA256", "SHA384" o "SHA512".
<code>encodingEncrypted</code>	texto	Codificación utilizada para convertir el mensaje binario encriptado en la cadena de resultados. Puede ser "Base64", o "Base64URL". Por defecto es "Base64".
<code>encodingDecrypted</code>	texto	Codificación utilizada para convertir el parámetro <code>mensaje</code> en la representación binaria a cifrar. Puede ser "UTF-8", "Base64" o "Base64URL". Por defecto es "UTF-8".

## Resultado

El valor devuelto es un mensaje encriptado.

## .getPrivateKey()

► Histórico

`.getPrivateKey() : Text`

Parámetros	Tipo		Descripción
Resultado	Texto	<-	Llave privada en formato PEM

La función `.getPrivateKey()` devuelve la llave privada del objeto `CryptoKey` en formato PEM, o una cadena vacía si no hay ninguna disponible.

#### *Resultado*

El valor devuelto es la llave privada.

## `.getPublicKey()`

► Histórico

`.getPublicKey( ) : Text`

Parámetros	Tipo		Descripción
Resultado	Texto	<-	Llave pública en formato PEM

La función `.getPublicKey()` devuelve la llave pública del objeto `CryptoKey` en formato PEM, o una cadena vacía si no hay ninguna disponible.

#### *Resultado*

El valor devuelto es la llave pública.

`---# # .pem`

► Histórico

`.pem : Text`

Definición PEM de una llave de encriptación a cargar<! -- END REF -->. Si la llave es una llave privada, se deducirá de ella la llave pública RSA o ECDSA.

## `.sign()`

► Histórico

`.sign (message : Text ; options : Text) : Text`

Parámetros	Tipo		Descripción
message	Texto	->	Cadena mensaje a firmar
options	Objeto	->	Opciones de firma
Resultado	Texto	<-	Firma en representación Base64 o Base64URL, según la opción "encoding"

La función `.sign()` firma la representación utf8 de un `message` string utilizando las llaves del objeto `CryptoKey` y las `options` suministradas. Devuelve su firma en formato base64 o base64URL, dependiendo del valor del atributo `options.encoding` que haya pasado.

`CryptoKey` debe contener una llave válida privada.

*options*

Propiedad	Tipo	Descripción
hash	texto	Algoritmo Digest a utilizar. Por ejemplo: "SHA256", "SHA384" o "SHA512". Cuando se utiliza para producir un JWT, el tamaño del hash debe coincidir con el tamaño del algoritmo PS@, ES@, RS@ o PS@
encodingEncrypted	texto	Codificación utilizada para convertir el mensaje binario encriptado en la cadena de resultados. Puede ser "Base64", o "Base64URL". Por defecto es "Base64".
pss	booleano	Utilice el Probabilistic Signature Scheme (PSS). Se ignora si la llave no es una llave RSA. Pase <code>true</code> al producir un JWT para el algoritmo PS@
encoding	texto	Representación que se utilizará para la firma de resultados. Los valores posibles son "Base64" o "Base64URL". Por defecto es "Base64".

## Resultado

`CryptoKey` debe contener una llave válida privada.

## .size

► Histórico

`.size : Integer`

Definido sólo para llaves RSA: el tamaño de la llave en bits. Normalmente 2048 (por defecto).

## .type

► Histórico

`.type : Text`

Nombre del tipo de llave - "RSA", "ECDSA", "PEM"

- "RSA": un par de llaves RSA, utilizando `settings.size` como `.size`.
- "ECDSA": un par de llaves del Algoritmo de Firma Digital de Curva Elíptica, utilizando `settings.curve` como `.curve`. Tenga en cuenta que las llaves ECDSA no pueden utilizarse para el cifrado, sino sólo para la firma.
- "PEM": una definición de par de llaves en formato PEM, utilizando `settings.pem` como `.pem`.

## .verify()

► Histórico

`.verify( message : Text ; signature : Text ; options : Object ) : object`

Parámetros	Tipo		Descripción
message	Texto	->	Cadena mensaje utilizada para generar la firma
signature	Texto	->	Firma a verificar, en representación Base64 o Base64URL, dependiendo del valor de <code>options.encoding</code>
options	Objeto	->	Opciones de firma
Resultado	Objeto	<-	Estado de la verificación

Representación utf8 de la cadena `message`.

La función `.verify()` verifica la firma base64 contra la representación utf8 del `message` utilizando las llaves del objeto `CryptoKey` y las `options` proporcionadas.

`options`

Propiedad	Tipo	Descripción
hash	texto	Algoritmo Digest a utilizar. Por ejemplo: "SHA256", "SHA384" o "SHA512". Cuando se utiliza para producir un JWT, el tamaño del hash debe coincidir con el tamaño del algoritmo PS@, ES@, RS@ o PS@
pss	booleano	Utilice el Probabilistic Signature Scheme (PSS). Se ignora si la llave no es una llave RSA. Pase <code>true</code> al verificar un JWT para el algoritmo PS@
encoding	texto	Representación de la firma facilitada. Puede ser "Base64" o "Base64URL". Por defecto es "Base64".

### Resultado

La `CryptoKey` debe contener una llave pública válida.

La función devuelve un objeto de estado con la propiedad `success` definida como `true` si el `message` pudo ser verificado con éxito (es decir, la firma coincide).

Propiedad	Tipo	Descripción
success	booleano	True si la firma coincide con el mensaje
errors	colección	Si <code>success</code> es <code>false</code> , puede contener una colección de errores

# DataClass

Una [DataClass](#) ofrece una interfaz de objeto a una tabla de la base de datos. Todas las clases de datos de una aplicación 4D están disponibles como una propiedad del `ds` [datastore](#).

## Resumen

`.attributeName : DataClassAttribute`

`.all ( { settings : Object } ) : 4D.EntitySelection`

consulta el datastore para encontrar todas las entidades relacionadas con la dataclass y las devuelve como una selección de entidades

`.clearRemoteCache()`

vacía la caché ORDA de una clase de datos

`.fromCollection( objectCol : Collection { ; settings : Object } ) : 4D.EntitySelection`

actualiza o crea entidades en la clase de datos según la colección de objetos `objectCol`, y devuelve la selección de entidades correspondiente

`.get( primaryKey : Integer { ; settings : Object } ) : 4D.Entity`

`.get( primaryKey : Text { ; settings : Object } ) : 4D.Entity`

consulta la clase de datos para recuperar la entidad que coincide con el parámetro `primaryKey`

`.getCount(): Integer`

devuelve el número de entidades de una clase de datos

`.getDataStore(): cs.DataStore`

`. getInfo() : Object`

devuelve un objeto que proporciona información sobre la clase de datos

`.getRemoteCache(): Objeto`

devuelve un objeto que contiene el contenido de la caché ORDA para una clase de datos

`.new() : 4D.Entity`

crea en memoria y devuelve una nueva entidad en blanco relacionada con la Dataclass

`.newSelection( { keepOrder : Integer } ) : 4D.EntitySelection`

creates a new, blank, non-shareable entity selection, related to the dataclass, in memory

`.query( queryString : Text { ; ...value : any } { ; querySettings : Object } ) : 4D.EntitySelection`

`.query( formula : Object { ; querySettings : Object } ) : 4D.EntitySelection`

`.setRemoteCacheSettings(settings: Object)`

sets the timeout and maximum size of the ORDA cache for a dataclass.

`.attributeName`

.attributeName : DataClassAttribute

## Descripción

Los atributos de las clases de datos son objetos que están disponibles directamente como propiedades de estas clases.

Los objetos devueltos son del tipo class [DataClassAttribute](#). Estos objetos tienen propiedades que puede leer para obtener información sobre los atributos de su clase de datos.

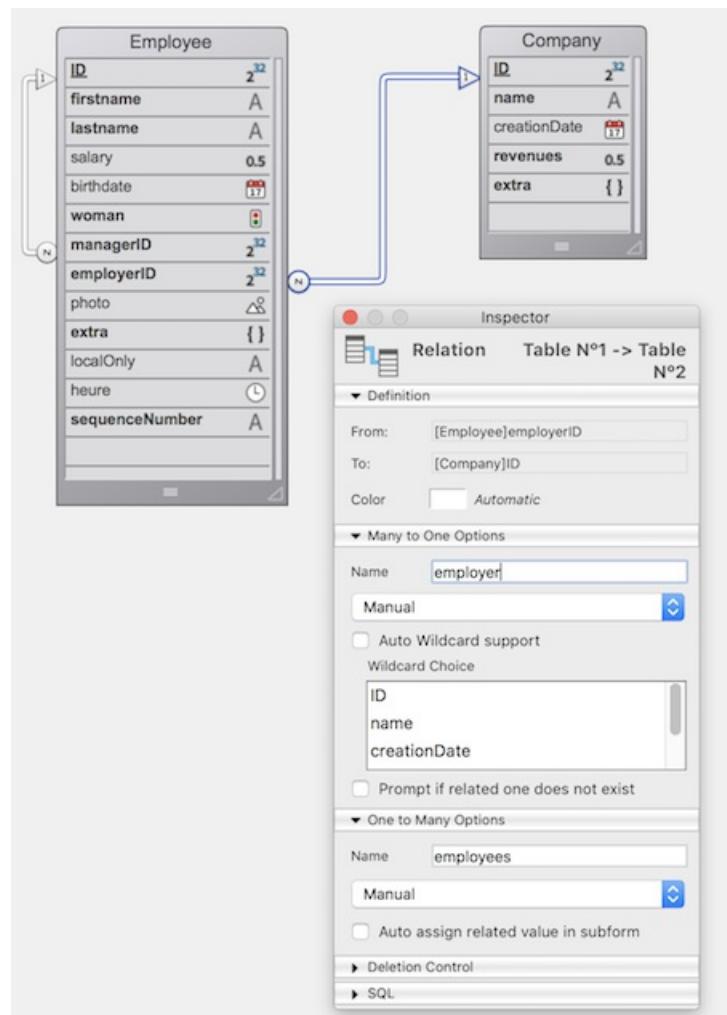
Los objetos del atributo Dataclass pueden ser modificados, pero la estructura subyacente de la base de datos no será alterada.

## Ejemplo 1

```
$salary:=ds.Employee.salary //devuelve el atributo salary en la clase de datos Employee  
$compCity:=ds.Company["city"] //devuelve el atributo city en la clase de datos Company
```

## Ejemplo 2

Considerando la siguiente estructura de la base:



```

var $firstnameAtt;$employerAtt;$employeesAtt : Object

$firstnameAtt:=ds.Employee.firstname
//{name:firstname,kind:storage,fieldType:0,type:string,fieldNumber:2,indexed:true,
//keyWordIndexed:false,autoFilled:false,mandatory:false,unique:false}

$employerAtt:=ds.Employee.employer
//{name:employer,kind:relatedEntity,relatedDataClass:Company,
//fieldType:38,type:Company,inverseName:employees}
//38=Is object

$employeesAtt:=ds.Company.employees
//{name:employees,kind:relatedEntities,relatedDataClass:Employee,
//fieldType:42,type:EmployeeSelection,inverseName:employer}
//42=Is collection

```

### Ejemplo 3

Considerando las propiedades de tabla siguientes:

The screenshot shows the 4D Studio Inspector window for the 'Employee' table. The 'sequenceNumber' field is selected. The 'Definition' tab shows the field is named 'sequenceNumber', has an Alpha type, and is set to 'Expose with 4D Mobile Service'. The 'Indexing' tab shows an 'Index' type of 'B-tree'. In the 'Data Entry Controls' tab, the 'Can't Modify' checkbox is checked.

```

var $sequenceNumberAtt : Object
$sequenceNumberAtt=ds.Employee.sequenceNumber
//{name:sequenceNumber,kind:storage,fieldType:0,type:string,fieldNumber:13,
//indexed:true,keyWordIndexed:false,autoFilled:true,mandatory:false,unique:true}

```

## .all()

► Histórico

.all ( { settings : Object } ) : 4D.EntitySelection

Parámetros	Tipo		Descripción
parámetros	Objeto	->	Opciones de construcción: context
Resultado	4D.EntitySelection	<-	Referencias sobre todas las entidades relacionadas con la clase de datos

### Descripción

La función `.all()` consulta el datastore para encontrar todas las entidades relacionadas con la dataclass y las devuelve como una selección de entidades.

Las entidades se devuelven en el orden por defecto, que es inicialmente el orden en que fueron creadas. Tenga en cuenta, sin embargo, que si se han eliminado entidades y se han añadido otras nuevas, el orden por defecto ya no refleja el orden de creación.

Si no se encuentra la entidad correspondiente, se devuelve una selección de entidades vacía.

Se aplica carga diferida.

parámetros

En el parámetro opcional *settings* se puede pasar un objeto que contenga opciones adicionales. Se soporta la siguiente propiedad:

Propiedad	Tipo	Descripción
context	Texto	Etiqueta para el contexto de optimización aplicado a la selección de entidades. Este contexto será utilizado por el código que maneja la selección de entidades para que pueda beneficiarse de la optimización. Esta funcionalidad está <a href="#">diseñada para el procesamiento cliente/servidor de ORDA</a> .

Para conocer el número total de entidades de una clase de datos, se recomienda utilizar la función `getCount()` más optimizada que la expresión `ds.myClass.all().length`.

Ejemplo

```
var $allEmp : cs.EmployeeSelection  
$allEmp:=ds.Employee.all()
```

## .clearRemoteCache()

► Histórico

.clearRemoteCache() | Parámetros | Tipo | | Descripción | | ----- | ---- |::| ----- | | | | | No requiere ningún parámetro |

Descripción

La función `.clearRemoteCache()` vacía la caché ORDA de una clase de datos.

Esta función no restablece los valores de `timeout` y `maxEntries`.

Ejemplo

```

var $ds : 4D.DataStoreImplementation
var $persons : cs.PersonsSelection
var $p : cs.PersonsEntity
var $cache : Object
var $info : Collection
var $text : Text

$ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")

$persons:=$ds.Persons.all()
$text:=""
For each ($p; $persons)
    $text:=$p.firstname+" lives in "+$p.address.city+ " "
End for each

$cache:=$ds.Persons.getRemoteCache()

$ds.Persons.clearRemoteCache()
// Caché de la dataclass Persons = {timeout:30;maxEntries:30000;stamp:255;entries:[]}

```

## .fromCollection()

► Histórico

.fromCollection( *objectCol* : Collection { ; *settings* : Object } ) : 4D.EntitySelection

Parámetros	Tipo		Descripción
<i>objectCol</i>	Collection	->	Colección de objetos a mapear con entidades
parámetros	Objeto	->	Opciones de construcción: context
Resultado	4D.EntitySelection	<-	Selección de entidades llenadas de la colección

### Descripción

La función `.fromCollection()` actualiza o crea entidades en la clase de datos según la colección de objetos *objectCol*, y devuelve la selección de entidades correspondiente.

En el parámetro *objectCol*, pasa una colección de objetos para crear nuevas entidades o actualizar las existentes de la clase de datos. Los nombres de las propiedades deben ser los mismos que los de los atributos de la clase de datos. Si un nombre de propiedad no existe en la clase de datos, se ignora. Si un valor de atributo no está definido en la colección, su valor es null.

El mapeo entre los objetos de la colección y las entidades se realiza sobre los nombres de atributos y tipos coincidentes. Si la propiedad de un objeto tiene el mismo nombre que el atributo de una entidad pero sus tipos no coinciden, el atributo de la entidad no se llena.

### Crear o actualizar modos

Para cada objeto de *objectCol*:

- Si el objeto contiene una propiedad booleana "\_\_NEW" establecida en false (o no contiene una propiedad booleana "\_\_NEW"), la entidad se actualiza o se crea con los valores correspondientes de las propiedades del objeto. No se realiza ninguna comprobación con respecto a la llave primaria:
  - Si la llave primaria se da y existe, la entidad se actualiza. En este caso, la llave primaria puede darse tal cual o con una propiedad "\_\_KEY" (llenada con el valor de la llave primaria).
  - Si se da la llave primaria (tal cual) y no existe, se crea la entidad
  - Si no se da la llave primaria, se crea la entidad y se asigna el valor de la llave primaria con respecto a las reglas estándar de la base de datos.
- Si el objeto contiene una propiedad booleana "\_\_NEW" establecida como true, la entidad se crea con los valores correspondientes de los atributos del objeto. Se realiza una verificación con respecto a la llave primaria:
  - Si se da la llave primaria (tal cual) y existe, se envía un error

- o Si se da la llave primaria (tal cual) y no existe, se crea la entidad
- o Si no se da la primaria, se crea la entidad y se asigna el valor de la llave primaria con respecto a las reglas estándar de la base de datos.

La propiedad "\_\_KEY" que contiene un valor sólo se tiene en cuenta cuando la propiedad "\_\_NEW" se establece como false (o se omite) y existe una entidad correspondiente. En todos los demás casos, el valor de la propiedad "\_\_KEY" se ignora, el valor de la llave primaria debe pasarse "tal cual".

## Related entities

Los objetos de *objectCol* pueden contener uno o más objetos anidados que presentan una o más entidades relacionadas, lo que puede ser útil para crear o actualizar enlaces entre entidades.

Los objetos anidados que presentan entidades relacionadas deben contener una propiedad "\_\_KEY" (llenada con el valor de la llave primaria de la entidad relacionada) o el atributo de llave primaria de la propia entidad relacionada. El uso de una propiedad \_\_KEY permite la independencia del nombre del atributo de la llave primaria.

El contenido de las entidades relacionadas no puede ser creado / actualizado a través de este mecanismo.

## Sello

Si se da un atributo \_\_STAMP, se realiza una comprobación con el sello en el almacén de datos y se puede devolver un error ("El sello dado no coincide con el actual para el registro# XX de la tabla XXXX"). Para más información, consulte [Entity locking](#).

## parámetros

En el parámetro opcional *settings* se puede pasar un objeto que contenga opciones adicionales. Se soporta la siguiente propiedad:

Propiedad	Tipo	Descripción
context	Texto	Etiqueta para el contexto de optimización aplicado a la selección de entidades. Este contexto será utilizado por el código que maneja la selección de entidades para que pueda beneficiarse de la optimización. Esta funcionalidad está <a href="#">diseñada para el procesamiento cliente/servidor de ORDA</a> .

## Ejemplo 1

Queremos actualizar una entidad existente. La propiedad \_\_NEW no se da, la llave primaria del empleado se da y existe:

```

var $empsCollection : Collection
var $emp : Object
var $employees : cs.EmployeeSelection

$empsCollection:=New collection
$emp:=New object
$emp.ID:=668 //Existing PK in Employee table
$emp.firstName:="Arthur"
$emp.lastName:="Martin"
$emp.employer:=New object("ID";121) //PK existente en la dataClass relacionada Company
// Para este empleado, podemos cambiar la Empresa utilizando otro PK existente en la dataClass relacio
$empsCollection.push($emp)
$employees:=ds.Employee.fromCollection($empsCollection)

```

## Ejemplo 2

Queremos actualizar una entidad existente. La propiedad \_\_NEW no se da, la llave primaria del empleado está con el atributo \_\_KEY y existe:

```

var $empsCollection : Collection
var $emp : Object
var $employees : cs.EmployeeSelection

$empsCollection:=New collection
$emp:=New object
$emp.__KEY:=1720 //PK existente en la tabla Employee
$emp.firstName:="John"
$emp.lastName:="Boorman"
$emp.employer:=New object("ID";121) // PK existente en la dataClass relacionada Company
// Para este empleado, podemos cambiar la Empresa utilizando otro PK existente en la dataClass relacio
$empsCollection.push($emp)
$employees:=ds.Employee.fromCollection($empsCollection)

```

### Ejemplo 3

Queremos crear simplemente una nueva entidad a partir de una colección:

```

var $empsCollection : Collection
var $emp : Object
var $employees : cs.EmployeeSelection

$empsCollection:=New collection
$emp:=New object
$emp.firstName:="Victor"
$emp.lastName:="Hugo"
$empsCollection.push($emp)
$employees:=ds.Employee.fromCollection($empsCollection)

```

### Ejemplo 4

Queremos crear una entidad. La propiedad \_\_NEW es True, la llave primaria del empleado no se da:

```

var $empsCollection : Collection
var $emp : Object
var $employees : cs.EmployeeSelection

$empsCollection:=New collection
$emp:=New object
$emp.firstName:="Mary"
$emp.lastName:="Smith"
$emp.employer:=New object("__KEY";121) //PK existente en la dataClass Company
$emp.__NEW:=True
$empsCollection.push($emp)
$employees:=ds.Employee.fromCollection($empsCollection)

```

### Ejemplo 5

Queremos crear una entidad. La propiedad \_\_NEW se omite, la llave primaria del empleado se da y no existe:

```

var $empsCollection : Collection
var $emp : Object
var $employees : cs.EmployeeSelection

$empsCollection:=New collection
$emp:=New object
$emp.ID:=10000 //Llave primaria inexistente
$emp.firstName:="Françoise"
$emp.lastName:="Sagan"
$empsCollection.push($emp)
$employees:=ds.Employee.fromCollection($empsCollection)

```

## Ejemplo 6

En este ejemplo, la primera entidad se creará y guardará pero la segunda fallará ya que ambas utilizan la misma llave primaria:

```

var $empsCollection : Collection
var $emp; $emp2 : Object
var $employees : cs.EmployeeSelection

$empsCollection:=New collection
$emp:=New object
$emp.ID:=10001 // Llave primaria inexistente
$emp.firstName:="Simone"
$emp.lastName:="Martin"
$emp.__NEW:=True
$empsCollection.push($emp)

$emp2:=New object
$emp2.ID:=10001 // La misma llave primaria, ya existente
$emp2.firstName:="Marc"
$emp2.lastName:="Smith"
$emp2.__NEW:=True
$empsCollection.push($emp2)
$employees:=ds.Employee.fromCollection($empsCollection)
//se crea la primera entidad
//error de llave duplicada para la segunda entidad

```

Ver también

[.toCollection\(\)](#)

## .get()

► Histórico

```

.get( primaryKey : Integer { ; settings : Object } ) : 4D.Entity
.get( primaryKey : Text { ; settings : Object } ) : 4D.Entity

```

Parámetros	Tipo		Descripción
primaryKey	Integer OR Text	->	Valor de la llave primaria de la entidad a recuperar
parámetros	Objeto	->	Opciones de construcción: context
Resultado	4D.Entity	<-	Entidad que coincide con la llave primaria designada

### Descripción

La función `.get()` consulta la clase de datos para recuperar la entidad que coincide con el parámetro `primaryKey`.

En *primaryKey*, pase el valor de la llave primaria de la entidad a recuperar. El tipo de valor debe coincidir con el tipo de la llave primaria definida en el almacén de datos (Entero o Texto). También puede asegurarse de que el valor de la llave primaria se devuelva siempre como Texto utilizando la función `.getKey()` con el parámetro `dk key as string`.

Si no se encuentra ninguna entidad con *primaryKey*, se devuelve una entidad Null.

Se aplica la carga diferida, lo que significa que los datos relacionados se cargan desde el disco sólo cuando son necesarios.

## parámetros

En el parámetro opcional *settings* se puede pasar un objeto que contenga opciones adicionales. Se soporta la siguiente propiedad:

Propiedad	Tipo	Descripción
context	Texto	Etiqueta para el contexto de optimización automática aplicado a la entidad. Este contexto será utilizado por el código siguiente que carga la entidad para que pueda beneficiarse de la optimización. Esta funcionalidad está <a href="#">diseñada para el procesamiento cliente/servidor de ORDA</a> .

## Ejemplo 1

```
var $entity : cs.EmployeeEntity
var $entity2 : cs.InvoiceEntity
$entity:=ds.Employee.get(167) // devuelve la entidad cuyo valor de llave primaria es 167
$entity2:=ds.Invoice.get("DGGX20030") // devuelve la entidad cuyo valor de llave primaria es "DGGX20030"
```

## Ejemplo 2

Este ejemplo ilustra el uso de la propiedad *context*:

```
var $e1; $e2; $e3; $e4 : cs.EmployeeEntity
var $settings; $settings2 : Object

$settings:=New object("context";"detail")
$settings2:=New object("context";"summary")

$e1:=ds.Employee.get(1;$settings)
completeAllData($e1) // En el método completeAllData, se lanza una optimización y se asocia al contexto

$e2:=ds.Employee.get(2;$settings)
completeAllData($e2) // En el método completeAllData, se aplica la optimización asociada al contexto "d

$e3:=ds.Employee.get(3;$settings2)
completeSummary($e3) //En el método completeSummary, se lanza una optimización y se asocia al contexto

$e4:=ds.Employee.get(4;$settings2)
completeSummary($e4) //En el método completeSummary se aplica la optimización asociada al contexto "sum
```

## .getCount()

► Histórico

`.getCount(): Integer`

Parámetros	Tipo		Descripción
result	Integer	<-	Número de entidades en la dataclass

## Descripción

La función `.getCount()` devuelve el número de entidades de una clase de datos.

Si se utiliza esta función dentro de una transacción, se tendrán en cuenta las entidades creadas durante la misma.

## Ejemplo

```
var $ds : 4D.DataStoreImplementation  
var $number : Integer  
  
$ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")  
  
$number:=$ds.Persons.getCount()
```

## .getDataStore()

► Histórico

`.getDataStore(): cs.DataStore`

Parámetros	Tipo		Descripción
Resultado	cs.DataStore	<-	Datastore de la dataclass

## Descripción

La función `.getDataStore()` devuelve el datastore para la clase de datos especificada.

El almacén de datos puede ser:

- el almacén de datos principal, devuelto por el comando `ds`.
- un almacén de datos remoto, abierto con el comando `Open datastore`.

## Ejemplo

El método de proyecto `SearchDuplicate` busca valores duplicados en cualquier clase de datos.

```
var $pet : cs.CatsEntity  
$pet:=ds.Cats.all().first() //obtener una entidad  
SearchDuplicate($pet;"Dogs")  
  
  
// método SearchDuplicate  
// SearchDuplicate(entity_to_search;dataclass_name)  
  
#DECLARE ($pet : Object ; $dataClassName : Text)  
var $dataStore; $duplicates : Object  
  
$dataStore:=$pet.getDataClass().getDataStore()  
$duplicates:=$dataStore[$dataClassName].query("name=:1";$pet.name)
```

## .getInfo()

► Histórico

`.getInfo() : Object`

Parámetros	Tipo		Descripción
Resultado	Objeto	<-	Información sobre la clase de datos

## Descripción

La función `.getInfo()` devuelve un objeto que proporciona información sobre la clase de datos. Esta función es útil para configurar el código genérico.

## Objeto devuelto

Propiedad	Tipo	Descripción
exposed	Booleano	True si la dataclass está expuesta en REST
name	Texto	Nombre de la dataclass
primaryKey	Texto	Nombre de la llave primaria de la clase de datos
tableNumber	Integer	Número de la tabla 4D interna

## Ejemplo 1

```
#DECLARE ($entity : Object)
var $status : Object

computeEmployeeNumber($entity) //realizar algunas acciones en la entidad

$status:=$entity.save()
if($status.success)
    ALERT("Record updated in table "+$entity.getDataClass().getInfo().name)
End if
```

## Ejemplo 2

```
var $settings : Object
var $es : cs.ClientsSelection

$settings:=New object
$settings.parameters:=New object("receivedIds";getIds())
$settings.attributes:=New object("pk";ds.Clients.getInfo().primaryKey)
$es:=ds.Clients.query(":pk in :receivedIds";$settings)
```

## Ejemplo 3

```
var $pk : Text
var $dataClassAttribute : Object

$pk:=ds.Employee.getInfo().primaryKey
$dataClassAttribute:=ds.Employee[$pk] // Si es necesario, el atributo que coincide con la llave primaria
```

## Ver también

[DataClassAttribute.exposed](#)

[`.getRemoteCache\(\)`](#)

► Histórico

## .getRemoteCache(): Objeto

Parámetros	Tipo		Descripción
result	Objeto	<-	Objeto que describe el contenido de la caché ORDA para la clase de datos.

Modo avanzado: esta función está pensada para los desarrolladores que necesitan personalizar las funcionalidades por defecto de ORDA para configuraciones específicas. En la mayoría de los casos, no será necesario utilizarla.

### Descripción

La función `.getRemoteCache()` devuelve un objeto que contiene el contenido de la caché ORDA para una clase de datos.

Llamar a esta función desde una aplicación monopuesto de 4D devuelve `Null`.

El objeto devuelto tiene las siguientes propiedades:

Propiedad	Tipo	Descripción
maxEntries	Integer	Número máximo de colecciones "entries".
stamp	Integer	Marcador de la caché.
timeout	Integer	Time remaining before the new entries in the cache are marked as expired.
entries	Collection	Contains an entry object for each entity in the cache.

Each entry object in the `entries` collection has the following properties:

Propiedad	Tipo	Descripción
data	Objeto	Object holding data on the entry.
expired	Booleano	True if the entry has expired.
key	Texto	Llave primaria de la entidad.

El objeto `data` de cada entrada contiene las siguientes propiedades:

Propiedad	Tipo	Descripción
<code>__KEY</code>	Cadena	Llave primaria de la entidad
<code>__STAMP</code>	Entero largo	Timestamp de la entidad en la base de datos
<code>__TIMESTAMP</code>	Cadena	Stamp of the entity in the database (format is YYYY-MM-DDTHH:MM:SS:ms:Z)
<code>dataClassNameAttributeName</code>	Variant	If there is data in the cache for a dataclass attribute, it is returned in a property with the same type as in the database.

Data concerning related entities is stored in the cache of the data object.

### Ejemplo

In the following example, `$ds.Persons.all()` loads the first entity with all its attributes. Then, the request optimization is triggered, so only `firstname` and `address.city` are loaded.

Note that `address.city` is loaded in the cache of the `Persons` dataclass.

Only the first entity of the `Address` dataclass is stored in the cache. It is loaded during the first iteration of the loop.

```

var $ds : 4D.DataStoreImplementation
var $persons : cs.PersonsSelection
var $p : cs.PersonsEntity
var $cachePersons; $cacheAddress : Object
var $text : Text

$ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")

$persons:=$ds.Persons.all()

$text:=""
For each ($p; $persons)
    $text:=$p.firstname+" lives in "+$p.address.city+ " / "
End for each

$cachePersons:=$ds.Persons.getRemoteCache()
$cacheAddress:=$ds.Adress.getRemoteCache()

```

## Ver también

[.setRemoteCacheSettings\(\)](#)  
[.clearRemoteCache\(\)](#)

## .new()

► Histórico

.new() : 4D.Entity

Parámetros	Tipo		Descripción
Resultado	4D.Entity	<-	Nueva entidad que coincide con la clase de datos

### Descripción

La función `.new()` crea en memoria y devuelve una nueva entidad en blanco relacionada con la Dataclass.

El objeto entidad se crea en memoria y no se guarda en la base de datos hasta que se llama a la función `.save( )`. Si la entidad se borra antes de ser guardada, no se puede recuperar.

4D Server: en cliente-servidor, si la llave primaria de la tabla correspondiente se autoincrementa, se calculará cuando la entidad se guarde en el servidor.

Todos los atributos de la entidad se inicializan con el valor null.

Los atributos se pueden inicializar con valores por defecto si se selecciona la opción Traducir los NULL a valores vacío al nivel de la estructura de la base 4D.

## Ejemplo

Este ejemplo crea una nueva entidad en la clase de datos "Log" y registra la información en el atributo "info":

```

var $entity : cs.LogEntity
$entity:=ds.Log.new() //crea una referencia
$entity.info:="New entry" //almacenar alguna información
$entity.save() //guardar la entidad

```

## .newSelection()

► Histórico

.newSelection( { keepOrder : Integer } ) : 4D.EntitySelection

Parámetros	Tipo		Descripción
keepOrder	Integer	->	dk keep ordered : crea una selección de entidades ordenada, dk no ordenada : crea una selección de entidades no ordenada (por defecto si se omite)
Resultado	4D.EntitySelection	<-	Nueva selección de entidades en blanco relacionadas con la clase de datos

## Descripción

The `.newSelection()` function creates a new, blank, non-shareable entity selection, related to the dataclass, in memory.

Para más información sobre las selecciones de entidades no compatibles, consulte [esta sección](#).

Si quieras crear una selección de entidades ordenada, pase el selector `dk keep ordered` en el parámetro `keepOrder`. Por defecto, si se omite este parámetro, o si se pasa el selector `dk non ordered`, el método crea una selección de entidades no ordenada. Las selecciones de entidades desordenadas son más rápidas pero no se puede confiar en las posiciones de las entidades. Para más información, consulte [Selecciones de entidades ordenadas y desordenadas](#).

Cuando se crea, la selección de entidades no contiene ninguna entidad (`mySelection.length` devuelve 0). Este método le permite crear selecciones de entidades gradualmente haciendo llamadas posteriores a la función `add()`.

## Ejemplo

```
var $USelection; $OSelection : cs.EmployeeSelection  
$USelection:=ds.Employee.newSelection() //crea una selección de entidades vacía y desordenada  
$OSelection:=ds.Employee.newSelection(dk keep ordered) //crea una selección de entidades vacía y ordenada
```

## .query()

► Histórico

.query( *queryString* : Text { ; ...*value* : any } { ; *querySettings* : Object } ) : 4D.EntitySelection

.query( *formula* : Object { ; *querySettings* : Object } ) : 4D.EntitySelection

Parámetros	Tipo		Descripción
<i>queryString</i>	Texto	->	Criterios de búsqueda como cadena
<i>formula</i>	Objeto	->	Criterios de búsqueda como objeto fórmula
<i>value</i>	any	->	Valor(es) a utilizar para los marcadores de posición indexados
<i>querySettings</i>	Objeto	->	Opciones de búsqueda: parameters, attributes, args, allowFormulas, context, queryPath, queryPlan
Resultado	4D.EntitySelection	<-	Nueva selección de entidades formada por las entidades de la clase de datos que cumplen los criterios de búsqueda especificados en <i>queryString</i> o <i>formula</i>

## Descripción

La función `.query()` busca entidades que cumplan con los criterios de búsqueda especificados en `queryString` o `formula` y (opcionalmente)`value(s)`, para todas las entidades de la clase de datos, y devuelve un nuevo objeto de tipo `EntitySelection` que contiene todas las entidades encontradas. Se aplica carga diferida.

Si no se encuentran entidades coincidentes, se devuelve una `EntitySelection` vacía.

parámetro `queryString`

El parámetro `queryString` utiliza la siguiente sintaxis:

```
attributePath|formula comparator value  
{logicalOperator attributePath|formula comparator value}  
{order by attributePath {desc | asc}}
```

donde:

- `attributePath`: ruta del atributo sobre el que se quiere ejecutar la búsqueda. Este parámetro puede ser un nombre simple (por ejemplo, "país") o cualquier ruta de atributo válida (por ejemplo, "país.nombre"). En el caso de una ruta de atributos de tipo `Collection`, se utiliza la notación `[ ]` para manejar todas las ocurrencias (por ejemplo "niños[ ].edad").

*No puede utilizar directamente atributos cuyo nombre contenga caracteres especiales como ".", "[ ]", o "=, ">, "#"... , porque se evaluarán incorrectamente en la cadena de consulta. Si necesita consultar dichos atributos, debe considerar el uso de marcadores, que permiten un rango ampliado de caracteres en las rutas de los atributos (ver Uso de marcadores de posición a continuación).*

- `formula`: una fórmula válida pasada como `Text` o en `Object`. La fórmula se evaluará para cada entidad procesada y debe devolver un valor booleano. Dentro de la fórmula, la entidad está disponible a través del objeto `This`.
  - `Text`: la cadena de fórmulas debe ir precedida de la declaración `eval( )`, para que el analizador de consultas evalúe la expresión correctamente. Por ejemplo: `"eval(length(This.lastname) >=30)"`
  - `Object`: el `objeto fórmula` se pasa como un marcador (ver abajo). La fórmula debe haber sido creada utilizando los comandos `Formula` o `Formula from string`.

- Tenga en cuenta que las fórmulas de 4D sólo admiten los símbolos `&` y `|` como operadores lógicos.
- Si la fórmula no es el único criterio de búsqueda, el optimizador del motor de búsquedas podría procesar previamente otros criterios (por ejemplo, los atributos indexados) y, por tanto, la fórmula podría evaluarse sólo para un subconjunto de entidades.

Las fórmulas en las consultas pueden recibir parámetros a través de `$1`. Este punto se detalla en el párrafo Parámetro fórmula más abajo.

- También puede pasar directamente un objeto parámetro `formula` en lugar del parámetro `queryString` (recomendado cuando las fórmulas son más complejas). Ver el párrafo Parámetro fórmula más abajo.
  - Por razones de seguridad, las llamadas a fórmulas dentro de las funciones `query()` pueden ser desestimadas. Ver la descripción del parámetro `querySettings`.
- `comparator`: símbolo que compara `attributePath` y `value`. Se soportan los siguientes símbolos:

Comparación	Símbolo(s)	Comentario
Igual a	=, ==	Obtiene los datos coincidentes, admite el comodín (@), no distingue entre mayúsculas de minúsculas ni diacríticas.
	==>, IS	Obtiene los datos coincidentes, considera @ como carácter estándar, no distingue entre mayúsculas de minúsculas ni diacríticas
Diferente de	#, !=	Soporta el comodín (@)
	!=>, IS NOT	Considera la @ como un carácter estándar
Menor que	<	
Mayor que	>	
Menor o igual que	<=	
Mayor o igual que	>=	
Incluído en	IN	Devuelve los datos iguales a al menos uno de los valores de una colección o de un conjunto de valores, admite el comodín (@)
Condición No aplicada a una sentencia	NOT	Los paréntesis son obligatorios cuando se utiliza NOT antes de una instrucción que contiene varios operadores
Contiene palabra clave	%	Las palabras claves pueden utilizarse en atributos de tipo texto o imagen

- value: el valor a comparar con el valor actual de la propiedad de cada entidad en la selección de entidades o elemento en la colección. Puede ser un marcador (ver Uso de marcadores más adelante) o cualquier expresión que coincida con la propiedad de tipo de datos.

Al utilizar un valor constante, deben respetarse las siguientes reglas:

- La constante de tipo texto puede pasarse con o sin comillas simples (ver Uso de comillas más abajo). Para consultar una cadena dentro de otra cadena (una consulta de tipo "contiene"), utilice el símbolo de comodín (@) en el valor para aislar la cadena a buscar como se muestra en este ejemplo: "@Smith@". Las siguientes palabras claves están prohibidas para las constantes de texto: true, false.
  - Valores constantes de tipo boolean: true o false (Sensible a las mayúsculas y minúsculas).
  - Valores constantes de tipo numérico: los decimales se separan con un '.'
  - Constantes de tipo date: formato "YYYY-MM-DD"
  - Constantes null: utilizando la palabra clave "null" encontrará las propiedades null y undefined.
  - en el caso de una búsqueda con un comparador IN, el valor debe ser una colección, o valores que coincidan con el tipo de la ruta del atributo entre [ ] separados por comas (para las cadenas, los caracteres " deben escaparse con \ ).
- logicalOperator: utilizado para unir condiciones múltiples en la búsqueda (opcional). Puede utilizar uno de los siguientes operadores lógicos (se puede utilizar el nombre o el símbolo):

Conjunción	Símbolo(s)
AND	&, &&, and
O	,   , or

- order by attributePath : puede incluir una declaración order by attributePath en la búsqueda para que los datos resultantes se ordenen de acuerdo con esa declaración. Puede utilizar varias instrucciones de ordenación, separadas por comas (por ejemplo, ordenación por attributePath1 desc, attributePath2 asc). Por defecto, el orden es ascendente. Pase 'desc' para definir un orden descendente y 'asc' para definir un orden ascendente.

Si utiliza esta declaración, la entity selection devuelta está ordenada (para más información, consulte [Entity selections ordenadas vs desordenadas](#)).

Cuando utilice comillas dentro de las consultas, debe utilizar comillas simples '' dentro de la consulta y comillas dobles "" para encerrar toda la consulta, de lo contrario se devuelve un error. Por ejemplo:

```
"employee.name = 'smith' AND employee.firstname = 'john'"
```

Las comillas simples ('') no se admiten en los valores buscados, ya que romperían la cadena de búsqueda. Por ejemplo, "comp.name = 'John's pizza' " generará un error. Si necesita buscar en valores con comillas simples, puede considerar el uso de marcadores de posición (ver más abajo).

## Uso del paréntesis

Puede utilizar paréntesis en la búsqueda para dar prioridad al cálculo. Por ejemplo, puede organizar una búsqueda de la siguiente manera:

```
"(employee.age >= 30 OR employee.age <= 65) AND (employee.salary <= 10000 OR employee.status = 'Manager'"
```

## Uso de marcadores de posición

4D le permite utilizar marcadores para los argumentos *attributePath*, *formula* y *value* dentro del parámetro *queryString*. Un marcador es un parámetro que se inserta en las cadenas de búsqueda y que se sustituye por otro valor cuando se evalúa la cadena de búsqueda consulta. El valor de los marcadores se evalúa una vez al principio de la búsqueda; no se evalúa para cada elemento.

Se pueden utilizar dos tipos de marcadores: indexed placeholders y los named placeholders:

	Marcadores de posición indexados	Nombre del marcador de posición
Definición	Los parámetros se insertan como <code>:paramIndex</code> (por ejemplo <code>:1, :2...</code> ) en <i>queryString</i> y sus valores correspondientes son suministrados por la secuencia de parámetros <i>value</i> . Puede utilizar hasta 128 parámetros <i>valor</i>	Los parámetros se insertan como <code>:paramName</code> (por ejemplo <code>:myparam</code> ) y sus valores se proporcionan en los atributos y/o objetos de parámetros en el parámetro <i>querySettings</i>
Ejemplo	<code>\$r:=class.query(":1=:2";"city";"Chicago")</code>	<code>\$o.attributes:=New object("att";"city") \$o.parameters:=New object("name";"Chicago") \$r:=class.query(":att=:name";\$o)</code>

Puede mezclar todos los tipos de argumentos en *queryString*. Una *queryString* puede contener, para los parámetros *attributePath*, *formula* y *value*:

- valores directos (sin marcadores),
- marcadores indexados y/o con nombre.

El uso de marcadores de posición en las búsquedas se recomienda por las siguientes razones:

1. Evita la inserción de código malicioso: si utiliza directamente variables completadas por el usuario dentro de la cadena de búsqueda, un usuario podría modificar las condiciones de búsqueda introduciendo argumentos de búsqueda adicionales. Por ejemplo, imagine una cadena de búsqueda como:

```
$vquery:="status = 'public' & name = "+myname //el usuario introduce su nombre  
$result:=$col.query($vquery)
```

Esta búsqueda parece segura ya que se filtran los datos no públicos. Sin embargo, si el usuario introduce en el área *myname* algo como *"smith OR status='private'*, la cadena de búsqueda se modificaría en la etapa de la interpretación y podría devolver datos privados.

Cuando se utilizan marcadores de posición, no es posible anular las condiciones de seguridad:

```
$result:=$col.query("status='public' & name=:1";myname)
```

En este caso, si el usuario introduce *smith OR status='private'* en el área *myname*, no se interpretará en la cadena de búsqueda, sino que sólo se pasará como valor. La búsqueda de una persona llamada "smith OR status='private'" simplemente fallará.

2. Evita tener que preocuparse por cuestiones de formato o caracteres, especialmente cuando se manejan los parámetros *attributePath* o *value* que pueden contener caracteres no alfanuméricos como ".", "[..."
3. Permite el uso de variables o expresiones en los argumentos de búsqueda. Ejemplos:

```
$result:=$col.query("address.city = :1 & name =:2;$city;$myVar+@")
$result2:=$col.query("company.name = :1;"John's Pizzas")
```

### Búsqueda de valores null

Cuando se buscan valores null, no se puede utilizar la sintaxis de marcador de posición porque el motor de búsqueda considera null como un valor de comparación invalido. Por ejemplo, si ejecuta la siguiente búsqueda:

```
$vSingles:=ds.Person.query("spouse = :1";Null) // NO funcionará
```

No obtendrá el resultado esperado porque el valor null será evaluado por 4D como un error resultante de la evaluación del parámetro (por ejemplo, un atributo procedente de otra búsqueda). Para este tipo de búsquedas, debe utilizar la sintaxis de búsqueda directa:

```
$vSingles:=ds.Person.query("spouse = null") // Sintaxis correcta
```

### Vinculación de los argumentos de búsqueda y los atributos de colección

Al buscar en colecciones dentro de los atributos de los objetos utilizando múltiples argumentos de búsqueda unidos por el operador AND, es posible que desee asegurarse de que sólo se devuelven las entidades que contienen elementos que coinciden con todos los argumentos, y no las entidades en las que los argumentos pueden encontrarse en diferentes elementos. Para ello, es necesario vincular los argumentos de la búsqueda a los elementos de colección, de modo que sólo se encuentren los elementos únicos que contengan argumentos vinculados.

Por ejemplo, con las dos entidades siguientes:

```
Entity 1:
ds. People.name: "martin"
ds. People.places:
{ "locations" : [ {
    "kind":"home",
    "city":"paris"
} ] }

Entity 2:
ds. People.name: "smith"
ds. People.places:
{ "locations" : [ {
    "kind":"home",
    "city":"lyon"
} , {
    "kind":"office",
    "city":"paris"
} ] }
```

Quiere encontrar personas con un tipo de ubicación "home" en la ciudad "paris". Si escribe:

```
ds.People.query("places.locations[].kind= :1 and places.locations[].city= :2";"home";"paris")
```

... la búsqueda devolverá "martin" y "smith" porque "smith" tiene un elemento "locations" cuyo "tipo" es "home" y un elemento "locations" cuya "ciudad" es "paris", aunque sean elementos diferentes.

Si quiere obtener sólo las entidades en las que los argumentos correspondientes están en el mismo elemento de colección, necesita enlazar los argumentos. Para enlazar los argumentos de búsqueda:

- Añada una letra entre los [] en la primera ruta a enlazar y repita la misma letra en todos los argumentos enlazados. Por ejemplo: `locations[a].city and locations[a].kind`. Puede utilizar cualquier letra del alfabeto latino (no diferencia entre mayúsculas y minúsculas).
- Para añadir diferentes criterios vinculados en la misma consulta, utilice otra letra. Puede crear hasta 26 combinaciones de criterios en una sola consulta.

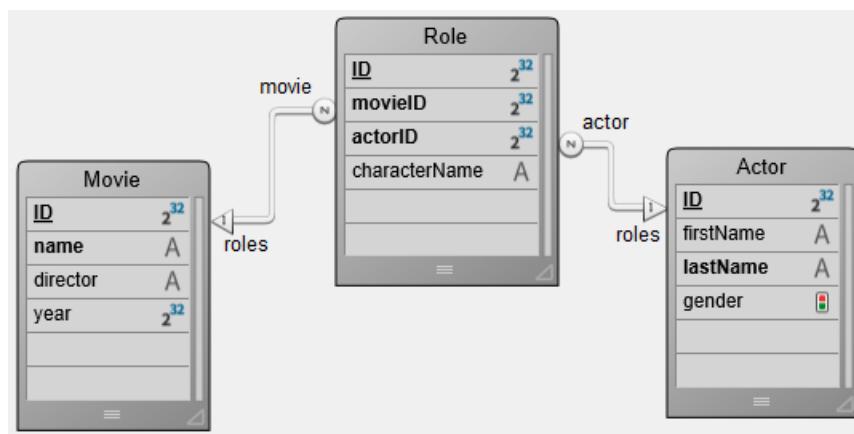
Con las entidades anteriores, si escribe:

```
ds.People.query("places.locations[a].kind= :1 and places.locations[a].city= :2";"home";"paris")
```

... la búsqueda sólo devolverá "martin" porque tiene un elemento "locations" cuyo "kind" es "home" y cuyo "city" es "paris". La búsqueda no devolverá "smith" porque los valores "home" y "paris" no están en el mismo elemento de colección.

Búsquedas en las relaciones muchos a muchos

ORDA ofrece una sintaxis especial para facilitar las consultas en las relaciones de muchos a muchos. En este contexto, puede ser necesario buscar diferentes valores con un operador AND PERO en el mismo atributo. Por ejemplo, de una mirada a la siguiente estructura:



Imagine que quiere buscar todas las películas en las que un actor A y un actor B tienen un papel. Si escribe una búsqueda simple utilizando un operador AND, no funcionará:

```
// código inválido
$es:=ds.Movie.query("roles.actor.lastName = :1 AND roles.actor.lastName = :2";"Hanks";"Ryan")
// $es está vacía
```

Básicamente, el problema está relacionado con la lógica interna de la búsqueda: no se puede buscar un atributo cuyo valor sea tanto "A" como "B".

To make it possible to perform such queries, ORDA allows a special syntax: you just need to add a *class index* between {} in all additional relation attributes used in the string:

```
"relationAttribute.attribute = :1 AND relationAttribute{x}.attribute = :2 [AND relationAttribute{y}.attr
```

{x} tells ORDA to create another reference for the relation attribute. It will then perform all the necessary bitmap operations internally. Note that x can be any number except 0: {1}, or {2}, or {1540}... ORDA only needs a unique reference in the query for each class index.

En nuestro ejemplo, sería:

```
// código válido
$es:=ds.Movie.query("roles.actor.lastName = :1 AND roles.actor{2}.lastName = :2";"Hanks";"Ryan")
// $es contiene los films (You've Got Mail, Sleepless in Seattle, Joe Versus the Volcano)
```

#### parámetro formula

Como alternativa a la inserción de fórmulas dentro del parámetro `queryString` (ver arriba), puede pasar directamente un objeto fórmula como criterio de búsqueda booleano. La utilización de un objeto fórmula para las búsquedas es recomendada ya que se beneficia de la tokenización, y el código es más fácil de buscar/leer.

La fórmula debe haber sido creada utilizando los comandos `Formula` o `Formula from string`. En este caso:

- `fórmula` se evalúa para cada entidad y debe devolver true o false. Durante la ejecución de la búsqueda, si el resultado de la fórmula no es un booleano, se considera como false.
- dentro de la `fórmula`, la entidad está disponible a través del objeto `This`.
- si el objeto `Formula` es null, se genera el error 1626 ("Esperando un texto o una fórmula"), que llama a interceptar utilizando un método instalado con `ON ERR CALL`.

Por razones de seguridad, las llamadas a fórmulas dentro de las funciones `query()` pueden ser desestimadas. Ver la descripción del parámetro `querySettings`.

#### Pasar parámetros a las fórmulas

Todo parámetro `formula` llamado por la función `query()` puede recibir parámetros:

- Los parámetros deben pasarse a través de la propiedad `args` (objeto) del parámetro `querySettings`.
- La fórmula recibe este objeto `args` como parámetro `$1`.

Este pequeño código muestra los principios de cómo se pasan los parámetros a los métodos:

```
$settings:=New object("args";New object("exclude";"-")) //objeto args a pasar los parámetros
$es:=ds.Students.query("eval(checkName($1.exclude));$settings") //args se recibe en $1
```

En el ejemplo 3 se ofrecen más ejemplos.

4D Server: en cliente/servidor, las fórmulas se ejecutan en el servidor. En este contexto, sólo se envía a las fórmulas el objeto `querySettings.args`.

#### Parámetro `querySettings`

En el parámetro `querySettings` se puede pasar un objeto que contenga opciones adicionales. Se soportan las siguientes propiedades:

Propiedad	Tipo	Descripción						
parameters	Objeto	Marcadores nombrados para los valores utilizados en <code>queryString</code> o <code>formula</code> . Los valores se expresan como pares propiedad / valor, donde propiedad es el nombre del marcador de posición insertado para un valor en <code>queryString</code> o <code>formula</code> (": <code>placeholder</code> ") y valor es el valor a comparar. Puede combinar marcadores de posición indexados (valores pasados directamente en parámetros de valor) y valores de marcadores de posición con nombre en la misma búsqueda.						
attributes	Objeto	Marcadores nombrados para las rutas de atributos utilizados en <code>queryString</code> o <code>formula</code> . Los atributos se expresan como pares propiedad / valor, donde propiedad es el nombre del marcador de posición insertado para una ruta de atributo en <code>queryString</code> o <code>formula</code> (": <code>placeholder</code> ") y valor puede ser una cadena o una colección de cadenas. Cada valor es una ruta que puede designar un escalar o un atributo relacionado de la dataclass o una propiedad en un campo de objeto de la dataclass <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Tipo</th><th>Descripción</th></tr> </thead> <tbody> <tr> <td>Cadena</td><td>attributePath expresado con la notación de puntos, por ejemplo: "name" o "user.address.zipCode"</td></tr> <tr> <td>Colección de cadenas</td><td>Cada cadena de la colección representa un nivel de attributePath, por ejemplo: ["name"] o ["user", "address", "zipCode"]. El uso de una colección permite consultar atributos con nombres que no se ajustan a la notación de puntos, por ejemplo, ["4Dv17.1", "en/fr"]</td></tr> </tbody> </table> Puede combinar marcadores de posición indexados (valores pasados directamente en los parámetros <code>value</code> ) y los valores de marcadores de posición con nombre en la misma búsqueda.	Tipo	Descripción	Cadena	attributePath expresado con la notación de puntos, por ejemplo: "name" o "user.address.zipCode"	Colección de cadenas	Cada cadena de la colección representa un nivel de attributePath, por ejemplo: ["name"] o ["user", "address", "zipCode"]. El uso de una colección permite consultar atributos con nombres que no se ajustan a la notación de puntos, por ejemplo, ["4Dv17.1", "en/fr"]
Tipo	Descripción							
Cadena	attributePath expresado con la notación de puntos, por ejemplo: "name" o "user.address.zipCode"							
Colección de cadenas	Cada cadena de la colección representa un nivel de attributePath, por ejemplo: ["name"] o ["user", "address", "zipCode"]. El uso de una colección permite consultar atributos con nombres que no se ajustan a la notación de puntos, por ejemplo, ["4Dv17.1", "en/fr"]							
args	Objeto	Parámetro(s) a pasar a las fórmulas, si las hay. El objeto <code>args</code> se recibirá en <code>\$1</code> dentro de las fórmulas y, por tanto, sus valores estarán disponibles a través de <code>\$1.property</code> (ver ejemplo 3).						
allowFormulas	Booleano	True para permitir las llamadas de fórmulas en la búsqueda (por defecto). Pase false para desautorizar la ejecución de fórmulas. Si se establece como false y <code>query()</code> recibe una fórmula, se envía un error (1278 - Fórmula no permitida en este método miembro).						
context	Texto	Etiqueta para el contexto de optimización automática aplicado a la entity selection. Este contexto será utilizado por el código que maneja la selección de entidades para que pueda beneficiarse de la optimización. Esta función está diseñada para el procesamiento cliente/servidor; para más información, consulte la sección Optimización cliente/servidor.						
queryPlan	Booleano	En la entity selection resultante, devuelve o no la descripción detallada de la búsqueda justo antes de que se ejecute, es decir, la búsqueda planificada. La propiedad devuelta es un objeto que incluye cada búsqueda y sub búsqueda prevista (en el caso de una búsqueda compleja). Esta opción es útil durante la fase de desarrollo de una aplicación. Suele utilizarse junto con <code>queryPath</code> . Por defecto si se omite: false. Nota: esta propiedad sólo la soportan las funciones <code>entitySelection.query()</code> y <code>dataClass.query()</code> .						
queryPath	Booleano	En la entity selection resultante, devuelve o no la descripción detallada de la búsqueda tal cual es realizada. La propiedad devuelta es un objeto que contiene la ruta utilizada para la búsqueda (normalmente idéntica a la de <code>queryPlan</code> , pero puede diferir si el motor consigue optimizar la búsqueda), así como el tiempo de procesamiento y el número de registros encontrados. Esta opción es útil durante la fase de desarrollo de una aplicación. Por defecto si se omite: false. Nota: esta propiedad sólo la soportan las funciones <code>entitySelection.query()</code> y <code>dataClass.query()</code> .						

## Sobre queryPlan y queryPath

La información registrada en `queryPlan` / `queryPath` incluye el tipo de búsqueda (indexada y secuencial) y cada

subconsulta necesaria junto con los operadores de conjunción. Las rutas de acceso de las peticiones también contienen el número de entidades encontradas y el tiempo necesario para ejecutar cada criterio de búsqueda. Las rutas de acceso de las peticiones también contienen el número de entidades encontradas y el tiempo necesario para ejecutar cada criterio de búsqueda. Generalmente, la descripción del plan de consulta y su ruta de acceso son idénticas, pero pueden diferir porque 4D puede implementar optimizaciones dinámicas cuando se ejecuta una consulta para mejorar el rendimiento. Por ejemplo, el motor 4D puede convertir dinámicamente una consulta indexada en una secuencial si estima que es más rápida. Este caso concreto puede darse cuando el número de entidades que se buscan es bajo.

Por ejemplo, si ejecuta la siguiente búsqueda:

```
$sel:=ds.Employee.query("salary < :1 and employer.name = :2 or employer.revenues > :3";\  
50000;"Lima West Kilo";10000000;New object("queryPath";True;"queryPlan";True))
```

queryPlan:

```
{Or:[{And:[{item:[index : Employee.salary ] < 50000},  
 {item:Join on Table : Company : Employee.employerID = Company. ID,  
 subquery:[{item:[index : Company.name ] = Lima West Kilo}]}]},  
 {item:Join on Table : Company : Employee.employerID = Company. ID,  
 subquery:[{item:[index : Company.revenues ] > 10000000}]}]}
```

queryPath:

```
{steps:[{description:OR,time:63,recordsfounds:1388132,  
 steps:[{description:AND,time:32,recordsfounds:131,  
 steps:[{description:[index : Employee.salary ] < 50000,time:16,recordsfounds:728260},{description:Jo  
 steps:[{steps:[{description:[index : Company.name ] = Lima West Kilo,time:0,recordsfounds:1}]}]}]},  
 steps:[{steps:[{description:[index : Company.revenues ] > 10000000,time:0,recordsfounds:933}]}]}]}
```

## Ejemplo 1

Esta sección ofrece varios ejemplos de búsquedas.

Consultas en una cadena:

```
$entitySelection:=ds.Customer.query("firstName = 'S@'")
```

Búsqueda con una instrucción NOT:

```
$entitySelection:=ds.Employee.query("not(firstName=Kim)")
```

Búsquedas con fechas:

```
$entitySelection:=ds.Employee.query("birthDate > :1";"1970-01-01")  
$entitySelection:=ds.Employee.query("birthDate <= :1";Current date-10950)
```

Búsqueda con marcadores de posición indexados para los valores:

```
$entitySelection:=ds.Customer.query("(firstName = :1 or firstName = :2) and (lastName = :3 or lastName =
```

Búsqueda con marcadores de posición indexados para valores en una dataclass relacionada:

```
$entitySelection:=ds.Employee.query("lastName = :1 and manager.lastName = :2";"M@";"S@")
```

Búsqueda con marcador de posición indexado que incluye una instrucción de orden descendente:

```
$entitySelection:=ds.Student.query("nationality = :1 order by campus.name desc, lastname";"French")
```

Búsqueda con marcadores de posición con nombre para los valores:

```
var $querySettings : Object  
var $managedCustomers : cs.CustomerSelection  
$querySettings:=New object  
$querySettings.parameters:=New object("userId";1234;"extraInfo";New object("name";"Smith"))  
$managedCustomers:=ds.Customer.query("salesperson.userId = :userId and name = :extraInfo.name";$querySet
```

Búsqueda que utiliza marcadores de posición con nombre e indexados para los valores:

```
var $querySettings : Object  
var $managedCustomers : cs.CustomerSelection  
$querySettings.parameters:=New object("userId";1234)  
$managedCustomers:=ds.Customer.query("salesperson.userId = :userId and name=:1";"Smith";$querySettings)
```

Búsqueda con objetos queryPlan y queryPath:

```
$entitySelection:=ds.Employee.query("(firstName = :1 or firstName = :2) and (lastName = :3 or lastName = :4);  
//puede obtener estas propiedades en la selección de entidades resultante  
var $queryPlan; $queryPath : Object  
$queryPlan:=$entitySelection.queryPlan  
$queryPath:=$entitySelection.queryPath
```

Búsqueda con una ruta de atributo de tipo Collection:

```
$entitySelection:=ds.Employee.query("extraInfo.hobbies[].name = :1";"horsebackriding")
```

Búsqueda con una ruta de atributos de tipo Collection y atributos vinculados:

```
$entitySelection:=ds.Employee.query("extraInfo.hobbies[a].name = :1 and extraInfo.hobbies[a].level=:2";"horsebackriding";2)
```

Búsqueda con una ruta de atributos de tipo Collection y múltiples atributos vinculados:

```
$entitySelection:=ds.Employee.query("extraInfo.hobbies[a].name = :1 and extraInfo.hobbies[a].level = :2 and extraInfo.hobbies[b].name = :3 and extraInfo.hobbies[b].level = :4";"horsebackriding";2;"Tennis";5)
```

Búsqueda con una ruta de atributo de tipo Objeto:

```
$entitySelection:=ds.Employee.query("extra.eyeColor = :1";"blue")
```

Búsqueda con una instrucción IN:

```
$entitySelection:=ds.Employee.query("firstName in :1";New collection("Kim";"Dixie"))
```

Búsqueda con instrucción NOT (IN):

```
$entitySelection:=ds.Employee.query("not (firstName in :1)";New collection("John";"Jane"))
```

Búsqueda con marcadores de posición indexados para los atributos:

```
var $es : cs.EmployeeSelection  
$es:=ds.Employee.query(":1 = 1234 and :2 = 'Smith'";"salesperson.userId";"name")  
//salesperson es una entidad relacionada
```

Búsqueda con marcadores de posición indexados para los atributos y marcadores de posición con nombre para los valores:

```
var $es : cs.EmployeeSelection  
var $querySettings : Object  
$querySettings:=New object  
$querySettings.parameters:=New object("customerName";"Smith")  
$es:=ds.Customer.query(":1 = 1234 and :2 = :customerName";"salesperson.userId";"name";$querySettings)  
//salesperson es una entidad relacionada
```

Búsqueda con marcadores de posición indexados para los atributos y los valores:

```
var $es : cs.EmployeeSelection  
$es:=ds.Clients.query(":1 = 1234 and :2 = :3";"salesperson.userId";"name";"Smith")  
//salesperson es una entidad relacionada
```

## Ejemplo 2

Esta sección ilustra las búsquedas con marcadores de posición con nombre para los atributos.

Dada una dataclass Employee con 2 entidades:

Entidad 1:

```
name: "Marie"  
number: 46  
softwares:{  
    "Word 10.2": "Installed",  
    "Excel 11.3": "To be upgraded",  
    "Powerpoint 12.4": "Not installed"  
}
```

Entidad 2:

```

name: "Sophie"
number: 47
softwares: {
  "Word 10.2": "Not installed",
  "Excel 11.3": "To be upgraded",
  "Powerpoint 12.4": "Not installed"
}

```

Búsqueda con marcadores de posición con nombre para los atributos:

```

var $querySettings : Object
var $es : cs.EmployeeSelection
$querySettings:=New object
$querySettings.attributes:=New object("attName";"name";"attWord";New collection("softwares";"Word 10.2"))
$es:=ds.Employee.query(":attName = 'Marie' and :attWord = 'Installed"';$querySettings)
//$es.length=1 (Employee Marie)

```

Búsqueda con marcadores de posición con nombre para los atributos y los valores:

```

var $querySettings : Object
var $es : cs.EmployeeSelection
var $name : Text
$querySettings:=New object
//Placeholders para los valores
//Se pide al usuario un nombre
$name:=Request("Por favor, introduzca el nombre a buscar:")
If(OK=1)
  $querySettings.parameters:=New object("givenName";$name)
//Placeholders para las rutas de atributos
  $querySettings.attributes:=New object("attName";"name")
  $es:=ds.Employee.query(":attName= :givenName";$querySettings)
End if

```

### Ejemplo 3

Estos ejemplos ilustran las distintas formas de utilizar fórmulas con o sin parámetros en sus búsquedas.

La fórmula se da como texto con `eval()` en el parámetro `queryString`:

```

var $es : cs.StudentsSelection
$es:=ds.Students.query("eval(length(This.lastname) >=30) and nationality='French'")

```

La fórmula se da como un objeto `Formula` a través de un marcador de posición:

```

var $es : cs.StudentsSelection
var $formula : Object
$formula:=Formula(Length(This.lastname)>=30)
$es:=ds.Students.query(":1 and nationality='French'";$formula)

```

Sólo se da como criterio un objeto `Formula`:

```

var $es : cs.StudentsSelection
var $formula : Object
$formula:=Formula(Length(This.lastname)>=30)
$es:=ds.Students.query($formula)

```

Se pueden aplicar varias fórmulas:

```

var $formula1; $1; $formula2 ;$0 : Object
$formula1:=$1
$formula2:=Formula(Length(This.firstname)>=30)
$0:=ds.Students.query(":1 and :2 and nationality='French'";$formula1;$formula2)

```

Una fórmula texto en *queryString* recibe un parámetro:

```

var $es : cs.StudentsSelection
var $settings : Object
$settings:=New object()
$settings.args:=New object("filter";"-")
$es:=ds.Students.query("eval(checkName($1.filter)) and nationality=:1";"French";$settings)

```

```

//checkName method
#DECLARE($exclude : Text) -> $result : Boolean
$result:=(Position($exclude;This.lastname)=0)

```

Utilizando el mismo método *checkName*, un objeto `Formula` como marcador de posición recibe un parámetro:

```

var $es : cs.StudentsSelection
var $settings; $formula : Object
$formula:=Formula(checkName($1.filter))
$settings:=New object()
$settings.args:=New object("filter";"-")
$es:=ds.Students.query(":1 and nationality=:2";$formula;"French";$settings)
$settings.args.filter:="*" // cambiar los parámetros sin actualizar el objeto $formula
$es:=ds.Students.query(":1 and nationality=:2";$formula;"French";$settings)

```

Queremos desautorizar las fórmulas, por ejemplo, cuando el usuario introduce su consulta:

```

var $es : cs.StudentsSelection
var $settings : Object
var $queryString : Text
$queryString:=Request("Enter your query:")
if(OK=1)
    $settings:=New object("allowFormulas";False)
    $es:=ds.Students.query($queryString;$settings) //Se produce un error si $queryString contiene una fó
End if

```

Ver también

`.query()` para selecciones de entidades

## .setRemoteCacheSettings()

► Histórico

.setRemoteCacheSettings(*settings*: Object)

Parámetros	Tipo		Descripción
parámetros	Objeto	->	Object that sets the timeout and maximum size of the ORDA cache for the dataclass.

Modo avanzado: esta función está pensada para los desarrolladores que necesitan personalizar las funcionalidades por defecto de ORDA para configuraciones específicas. En la mayoría de los casos, no será necesario utilizarla.

## Descripción

The `.setRemoteCacheSettings()` function sets the timeout and maximum size of the ORDA cache for a dataclass..

In the *settings* parameter, pass an object with the following properties:

Propiedad	Tipo	Descripción
timeout	Integer	Tiempo de espera en segundos.
maxEntries	Integer	Número máximo de entidades.

`timeout` sets the timeout of the ORDA cache for the dataclass (default is 30 seconds). Once the timeout has passed, the entities of the dataclass in the cache are considered as expired. Esto significa que:

- los datos siguen estando ahí
- the next time the data is needed, it will be asked to the server
- 4D automatically removes expired data when the maximum number of entities is reached

Setting a `timeout` property sets a new timeout for the entities already present in the cache. It is useful when working with data that does not change very frequently, and thus when new requests to the server are not necessary.

`maxEntries` sets the max number of entities in the ORDA cache. Por defecto es 30 000.

The minimum number of entries is 300, so the value of `maxEntries` must be equal to or higher than 300. Otherwise it is ignored and the maximum number of entries is set to 300.

If no valid properties are passed as `timeout` and `maxEntries`, the cache remains unchanged, with its default or previously set values.

When an entity is saved, it is updated in the cache and expires once the timeout is reached.

## Ejemplo

```
var $ds : 4D.DataStoreImplementation  
  
$ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")  
  
$ds.Buildings.setRemoteCacheSettings(New object("timeout"; 60; "maxEntries"; 350))
```

## Ver también

[.clearRemoteCache\(\)](#)  
[.getRemoteCache\(\)](#)

# DataClassAttribute

Los atributos Dataclass están disponibles como propiedades de sus respectivas clases. Por ejemplo:

```
nameAttribute:=ds.Company.name //referencia a un atributo de clase  
revenuesAttribute:=ds.Company["revenues"] //método alternativo
```

Este código asigna a *nameAttribute* y *revenuesAttribute* referencias a los atributos name y revenues de la clase Company. Esta sintaxis NO devuelve los valores mantenidos dentro del atributo, sino que devuelve referencias a los propios atributos. Para manejar los valores, es necesario pasar por [Entidades](#).

Los objetos `DataClassAttribute` tienen propiedades que puede leer para obtener información sobre los atributos de su clase de datos.

Los objetos del atributo Dataclass pueden ser modificados, pero la estructura subyacente de la base de datos no será alterada.

## Resumen

<b>.autoFilled : Boolean</b>	contiene True si el valor del atributo es llenado automáticamente por 4D
<b>.exposed : Boolean</b>	"true" si el atributo está "expuesto" en REST
<b>.fieldNumber : Integer</b>	contiene el número de campo 4D interno del atributo
<b>.FieldType : Integer</b>	contiene el tipo de base de datos 4D del atributo
<b>.indexed : Boolean</b>	contiene True si hay un índice B-tree o Cluster B-tree en el atributo
<b>.inverseName: Texto</b>	devuelve el nombre del atributo que está al otro lado de la relación
<b>.keywordIndexed: Boolean</b>	contiene True si hay un índice de palabras clave en el atributo
<b>.kind : Text</b>	devuelve la categoría del atributo
<b>.mandatory : Boolean</b>	contiene True si se rechaza la entrada de valores Null para el atributo
<b>.name : Text</b>	devuelve el nombre del objeto <code>dataClassAttribute</code> como cadena
<b>.path : Text</b>	devuelve el nombre de un atributo alias basado en una relación
<b>.readOnly : Boolean</b>	"true" si el atributo está en solo lectura
<b>.relatedDataClass: Text</b>	devuelve el nombre de la clase de datos relacionada con el atributo
<b>.type : Text</b>	contiene el tipo de valor conceptual del atributo
<b>.unique : Boolean</b>	contiene True si el valor del atributo debe ser único

## .autoFilled

► Histórico

`.autoFilled : Boolean`

### Descripción

La propiedad `.autoFilled` contiene True si el valor del atributo es llenado automáticamente por 4D. Esta propiedad corresponde a las siguientes propiedades de campo 4D:

- "Autoincremento", para campos de tipo numérico

- "Auto UUID", para los campos UUID (tipo alfa).

Esta propiedad no se devuelve si `.kind` = "relatedEntity" o "relatedEntities".

Para la programación genérica, se puede utilizar `Bool(dataClassAttribute.autoFilled)` para obtener un valor válido (false) aunque no se devuelva `.autoFilled`.

## .exposed

► Histórico

`.exposed` : Boolean

### Descripción

La propiedad `.exposed` es "true" si el atributo está "expuesto" en REST.

### Ver también

[DataClass.getInfo\(\)](#)

## .fieldNumber

► Histórico

`.fieldNumber` : Integer

### Descripción

La propiedad `.fieldNumber` contiene el número de campo 4D interno del atributo.

Esta propiedad no se devuelve si `.kind` = "relatedEntity" o "relatedEntities".

Para la programación genérica, puede utilizar `Num(dataClassAttribute.fieldNumber)` para obtener un valor válido (0) aunque no se devuelva `.fieldNumber`.

## .fieldType

► Histórico

`.fieldType` : Integer

### Descripción

La propiedad `.fieldType` contiene el tipo de base de datos 4D del atributo. Depende del tipo de atributo (ver [.kind](#)).

Valores posibles:

dataClassAttribute.kind	fieldType
storage	Tipo de campo 4D correspondiente, ver <a href="#">Value type</a>
relatedEntity	38 ( Is object )
relatedEntities	42 ( Is collection )
calculated	<ul style="list-style-type: none"> <li>escalar: tipo de campo 4D correspondiente, ver <a href="#">Value type</a></li> <li>entity: 38 ( Is object )</li> <li>entity selection: 42 ( Is collection )</li> </ul>
alias	<ul style="list-style-type: none"> <li>escalar: tipo de campo 4D correspondiente, ver <a href="#">Value type</a></li> <li>entity: 38 ( Is object )</li> <li>entity selection: 42 ( Is collection )</li> </ul>

Ver también

[.type](#)

## .indexed

► Histórico

.indexed : Boolean

Descripción

La propiedad `.indexed` contiene True si hay un índice B-tree o Cluster B-tree en el atributo.

Esta propiedad no se devuelve si `.kind = "relatedEntity"` o `"relatedEntities"`.

Para la programación genérica, se puede utilizar `Bool(dataClassAttribute.indexed)` para obtener un valor válido (false) aunque no se devuelva `.indexed`.

## .inverseName

► Histórico

.inverseName: Texto

Descripción

La propiedad `.inverseName` devuelve el nombre del atributo que está al otro lado de la relación.

Esta propiedad no se devuelve si `.kind = "storage"`. Debe ser del tipo "relatedEntity" o "relatedEntities".

Para la programación genérica, puede utilizar `String(dataClassAttribute.inverseName)` para obtener un valor válido ("") aunque no se devuelva `.inverseName`.

## .keywordIndexed

► Histórico

.keywordIndexed: Boolean

Descripción

La propiedad `.keywordIndexed` contiene True si hay un índice de palabras clave en el atributo.

Esta propiedad no se devuelve si `.kind` = "relatedEntity" o "relatedEntities".

Para la programación genérica, se puede utilizar `Bool(dataClassAttribute.keywordIndexed)` para obtener un valor válido (false) aunque no se devuelva `.keywordIndexed`.

## .kind

► Histórico

`.kind` : Text

### Descripción

La propiedad `.kind` devuelve la categoría del atributo. El valor devuelto puede ser uno de los siguientes:

- "storage": atributo de almacenamiento (o escalar), es decir, atributo que almacena un valor, no una referencia a otro atributo
- "calculated": atributo calculado, es decir definido por [la función `get`](#)
- "alias": atributo basado en [otro atributo](#)
- "relatedEntity": N -> 1 atributo de relación (referencia a una entidad)
- "relatedEntities": 1 -> N atributo de relación (referencia a una selección de entidades)

### Ejemplo

Dada la siguiente tabla y relación:

```
var $attKind : Text
$attKind:=ds.Employee.lastname.kind // $attKind="storage"
$attKind:=ds.Employee.manager.kind // $attKind="relatedEntity"
$attKind:=ds.Employee.directReports.kind // $attKind="relatedEntities"
```

## .mandatory

► Histórico

`.mandatory` : Boolean

### Descripción

La propiedad `.mandatory` contiene True si se rechaza la entrada de valores Null para el atributo.

Esta propiedad no se devuelve si `.kind` = "relatedEntity" o "relatedEntities".

Para la programación genérica, se puede utilizar `Bool(dataClassAttribute.mandatory)` para obtener un valor válido (false) aunque no se devuelva `.mandatory`. Atención: esta propiedad corresponde a la propiedad del campo "Reject NULL value input" a nivel de la base de datos 4D. No tiene relación con la propiedad "Obligatorio" existente, que es una opción de control de entrada de datos para una tabla.

## .name

► Histórico

`.name` : Text

### Descripción

La propiedad `.name` devuelve el nombre del objeto `dataClassAttribute` como cadena.

## Ejemplo

```
var $attName : Text  
$attName:=ds.Employee.lastname.name //attName="lastname"
```

## .path

► Histórico

.path : Text

### Descripción

La propiedad `.path` devuelve el nombre de un atributo alias basado en una relación.

## Ejemplo

```
var $path : Text  
$path:=ds.Teacher.students.path //path="courses.student"
```

## .readOnly

► Histórico

.readOnly : Boolean

### Descripción

La propiedad `.readOnly` es "true" si el atributo está en solo lectura.

Por ejemplo, los atributos calculados sin función `set` son solo lectura.

## .relatedDataClass

► Histórico

.relatedDataClass: Text

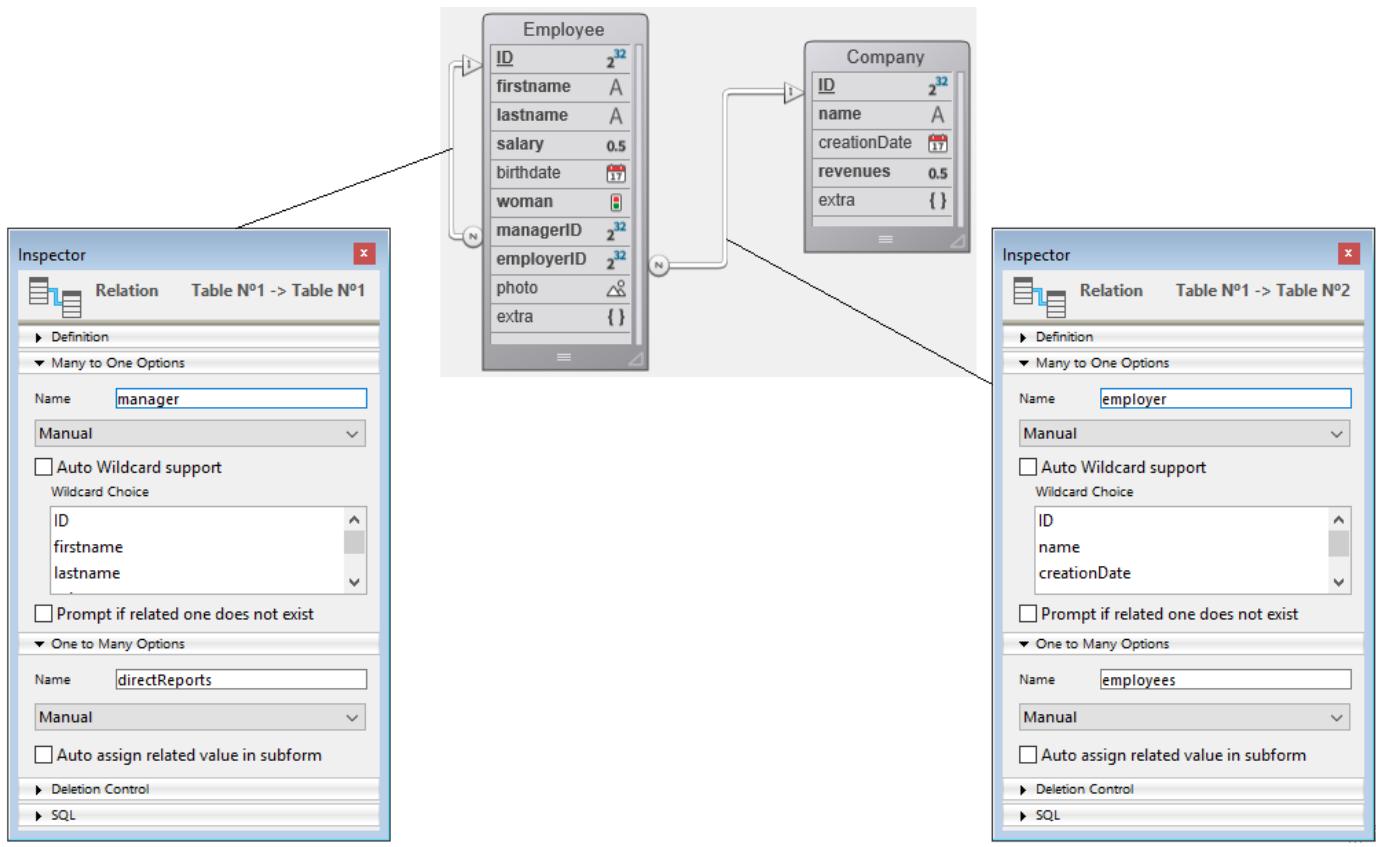
### Descripción

Esta propiedad sólo está disponible con atributos de la propiedad "relatedEntity" o "relatedEntities" `.kind`.

La propiedad `.relatedDataClass` devuelve el nombre de la clase de datos relacionada con el atributo.

## Ejemplo

Dadas las siguientes tablas y relaciones:



```
var $relClass1; $relClassN : Text
$relClass1:=ds.Employee.employer.relatedDataClass // $relClass1="Company"
$relClassN:=ds.Employee.directReports.relatedDataClass // $relClassN="Employee"
```

## .type

► Histórico

.type : Text

### Descripción

La propiedad `.type` contiene el tipo de valor conceptual del atributo, útil para la programación genérica.

El tipo de valor conceptual depende del atributo `.kind`.

Valores posibles:

dataClassAttribute.kind	type	Comentario
storage	"blob", "bool", "date", "image", "number", "object", o "string"	"number" se devuelve para cualquier tipo numérico, incluyendo la duración. "string" se devuelve para los tipos de campo uuid, alfa y texto. Los atributos "blob" son <a href="#">objetos blob</a> , se gestionan utilizando <a href="#">la clase Blob</a> .
relatedEntity	related dataClass name	Ej.: "Companies"
relatedEntities	related dataClass name + "Selection" suffix	Ej.: "EmployeeSelection"
calculated	<ul style="list-style-type: none"> <li>● storage: type ("blob", "number", etc.)</li> <li>● entity: dataClass name</li> <li>● entity selection: dataClass name + "Selection"</li> </ul>	

Ver también

[.fieldType](#)

## .unique

► Histórico

.unique : Boolean

### Descripción

La propiedad `.unique` contiene True si el valor del atributo debe ser único. Esta propiedad corresponde a la propiedad de campo 4D "Unique".

Esta propiedad no se devuelve si `.kind` = "relatedEntity" o "relatedEntities".

Para la programación genérica, se puede utilizar `Bool(dataClassAttribute.unique)` para obtener un valor válido (false) aunque no se devuelva `.unique`.

# DataStore

Un [Datastore](#) es el objeto de interfaz suministrado por ORDA para referenciar y acceder a una base de datos. Los objetos [Datastore](#) son devueltos por los siguientes comandos:

- [ds](#): un acceso directo al almacén de datos principal
- [Open datastore](#): para abrir todo almacén de datos remoto

## Resumen

<a href="#">.cancelTransaction()</a>	cancela la transacción
<a href="#">.clearAllRemoteContexts()</a>	clears all the attributes for all the active contexts in the datastore
<a href="#">.dataclassName : 4D.DataClass</a>	contiene una descripción de la clase de datos
<a href="#">.encryptionStatus(): Object</a>	devuelve un objeto que proporciona el estado de encriptación del archivo de datos actual
<a href="#">.getAllRemoteContexts() : Collection</a>	returns a collection of objects containing information on all the active optimization contexts in the datastore
<a href="#">.getInfo(): Object</a>	devuelve un objeto que proporciona información sobre el datastore
<a href="#">.getRemoteContextInfo(contextName : Text): Object</a>	returns an object that holds information on the <i>contextName</i> optimization context in the datastore.
<a href="#">.getRequestLog() : Collection</a>	devuelve las peticiones ORDA registradas en memoria del lado del cliente
<a href="#">.makeSelectionsAlterable()</a>	establece todas las selecciones de entidades como alterables por defecto en los datastores de la aplicación actual
<a href="#">.provideDataKey( curPassPhrase : Text ) : Object</a> <a href="#">.provideDataKey( curDataKey : Object ) : Object</a>	permite suministrar una llave de cifrado de datos para el archivo de datos actual del datastore y detecta si la llave coincide con los datos cifrados
<a href="#">.setAdminProtection( status : Boolean )</a>	permite deshabilitar cualquier acceso a datos en el <a href="#">puerto de administración web</a> , incluso para el <a href="#">Explorador de datos</a> en las sesiones de <a href="#">WebAdmin</a>
<a href="#">.setRemoteContextInfo( contextName : Text ; dataClassName : Text ; attributes : Text {; contextType : Text { ; pageLength : Integer}})</a> <a href="#">.setRemoteContextInfo( contextName : Text ; dataClassName : Text; attributesColl : Collection {; contextType : Text { ; pageLength : Integer }} )</a> <a href="#">.setRemoteContextInfo( contextName : Text ; dataClassObject : 4D.DataClass ; attributes : Text {; contextType : Text { ; pageLength : Integer }})</a> <a href="#">.setRemoteContextInfo( contextName : Text ; dataClassObject : 4D.DataClass ; attributesColl : Collection {; contextType : Text { ; pageLength : Integer }} )</a>	links the specified dataclass attributes to the <i>contextName</i> optimization context

<code>.startRequestLog()</code>	
<code>.startRequestLog( file : 4D.File )</code>	
<code>.startRequestLog( reqNum : Integer )</code>	inicia el registro de las peticiones ORDA del lado del cliente
<code>.startTransaction()</code>	inicia una transacción en el proceso actual en la base de datos que coincide con el datastore al que se aplica
<code>.stopRequestLog()</code>	detiene cualquier registro de peticiones ORDA del lado del cliente
<code>.validateTransaction()</code>	acepta la transacción

## ds

► Histórico

`ds { ( /localID : Text ) } : cs.DataStore`

Parámetros	Tipo		Descripción
<code>localID</code>	Texto	->	ID local del almacén de datos remoto a devolver
Resultado	<code>cs.DataStore</code>	<-	Referencia al almacén de datos

### Descripción

El comando `ds` devuelve una referencia al almacén de datos que coincide con la base de datos 4D actual o con la base de datos designada por `localID`.

Si se omite el parámetro `localID` (o se pasa una cadena vacía ""), el comando devuelve una referencia al almacén de datos que coincide con la base de datos local de 4D (o la base de datos de 4D Server en caso de abrir una base de datos remota en 4D Server). El almacén de datos se abre automáticamente y está disponible directamente a través de `ds`.

También puede obtener una referencia en un datastore remoto abierto pasando su id local en el parámetro `localID`. El almacén de datos debe haber sido abierto previamente con el comando `Open datastore` por la base de datos actual (local o componente). La identificación local se define cuando se utiliza este comando.

El alcance del id local es la base de datos en la que se ha abierto el almacén de datos.

Si no se encuentra ningún almacén de datos `localID`, el comando devuelve Null.

Objetos disponibles en `cs.Datastore` se mapean desde la base de datos de destino con respecto a las [reglas generales ORDA](#).

### Ejemplo 1

Utilizando el almacén de datos principal de la base 4D:

```
$result:=ds.Employee.query("firstName = :1";"S@")
```

### Ejemplo 2

```

var $connectTo; $firstFrench; $firstForeign : Object
var $frenchStudents; $foreignStudents : cs.DataStore

$connectTo:=New object("type";"4D Server";"hostname";"192.168.18.11:8044")
$frenchStudents:=Open datastore($connectTo;"french")

$connectTo.hostname:="192.168.18.11:8050"
$foreignStudents:=Open datastore($connectTo;"foreign")
//...
//...
$firstFrench:=getFirst("french";"Students")
$firstForeign:=getFirst("foreign";"Students")

```

```

//getFirst method
//getFirst(localID;dataclass) -> entity
#DECLARE( $localID : Text; $dataClassName : Text ) -> $entity : 4D.Entity

$0:=ds($localId)[$dataClassName].all().first()

```

## Open datastore

► Histórico

`Open datastore( connectionInfo : Object ; localID : Text ) : cs.DataStore`

Parámetros	Tipo		Descripción
connectionInfo	Objeto	->	Propiedades de conexión utilizadas para alcanzar el almacén de datos remoto
localID	Texto	->	Id para asignar al almacén de datos abierto en la aplicación local (obligatorio)
Resultado	cs.DataStore	<-	Objeto del almacén de datos

### Descripción

El comando `Open datastore` conecta la aplicación a la base de datos 4D identificada por el parámetro `connectionInfo` y devuelve un objeto `cs.DataStore` asociado al alias local `localID`.

La base de datos `connectionInfo` 4D debe estar disponible como almacén de datos remoto, es decir:

- su servidor web debe ser lanzado con http y/o https activado,
- su opción [Exponer como servidor REST](#) debe estar marcada,
- se dispone de al menos una licencia cliente.

Si no se encuentra ninguna base de datos coincidente, `Open datastore` devuelve Null.

`localID` es un alias local para la sesión abierta en el almacén de datos remoto. Si `localID` ya existe en la aplicación, se utiliza. En caso contrario, se crea una nueva sesión `localID` cuando se utiliza el objeto `datastore`.

Objetos disponibles en `cs.Datastore` se mapean desde la base de datos de destino con respecto a las [reglas generales ORDA](#).

Una vez abierta la sesión, las siguientes sentencias son equivalentes y devuelven una referencia sobre el mismo objeto `datastore`:

```

$myds:=Open datastore(connectionInfo;"myLocalId")
$myds2:=ds("myLocalId")
//$myds and $myds2 are equivalent

```

Pase en *connectionInfo* un objeto que describa el almacén de datos remoto al que desea conectarse. Puede contener las siguientes propiedades (todas las propiedades son opcionales excepto *hostname*):

Propiedad	Tipo	Descripción
hostname	Texto	Nombre o dirección IP de la base de datos remota + ":" + número de puerto (el número de puerto es obligatorio)
user	Texto	Nombre de usuario
contraseña	Texto	Contraseña del usuario
idleTimeout	Entero largo	Tiempo de espera de la sesión de inactividad (en minutos), después del cual la sesión es cerrada automáticamente por 4D. Si se omite, el valor por defecto es 60 (1h). El valor no puede ser < 60 (si se pasa un valor inferior, el tiempo de espera se establece en 60). Para más información, consulte Cierre de sesiones.
tls	Booleano	Utilice una conexión segura(*). Si se omite, es false por defecto. Se recomienda utilizar una conexión segura siempre que sea posible.
type	Texto	Debe ser "4D Server"

(\*) Si *tls* es true, se utiliza el protocolo HTTPS si:

- HTTPS está activado en el almacén de datos remoto
- el número de puerto especificado coincide con el puerto HTTPS configurado en los ajustes de la base de datos
- un certificado válido y una llave privada de encriptación están instalados en la base de datos. En caso contrario, se produce el error "1610 - Una solicitud remota al host xxx ha fallado"

## Ejemplo 1

Conexión a un almacén de datos remoto sin usuario/contraseña:

```
var $connectTo : Object
var $remoteDS : cs.DataStore
$connectTo:=New object("type";"4D Server";"hostname";"192.168.18.11:8044")
$remoteDS:=Open datastore($connectTo;"students")
ALERT("This remote datastore contains "+String($remoteDS.Students.all().length)+" students")
```

## Ejemplo 2

Conexión a un almacén de datos remoto con usuario/contraseña/ timeout / tls:

```
var $connectTo : Object
var $remoteDS : cs.DataStore
$connectTo:=New object("type";"4D Server";"hostname";"\\"192.168.18.11:4443";\
"user";"marie";"password";$pwd;"idleTimeout";70;"tls";True)
$remoteDS:=Open datastore($connectTo;"students")
ALERT("This remote datastore contains "+String($remoteDS.Students.all().length)+" students")
```

## Ejemplo 3

Trabajar con varios almacenes de datos remotos:

```

var $connectTo : Object
var $frenchStudents; $foreignStudents : cs.DataStore
$connectTo:=New object("hostname";"192.168.18.11:8044")
$frenchStudents:=Open datastore($connectTo;"french")
$connectTo.hostname:="192.168.18.11:8050"
$foreignStudents:=Open datastore($connectTo;"foreign")
ALERT("They are "+String($frenchStudents.Students.all().length)+" French students")
ALERT("They are "+String($foreignStudents.Students.all().length)+" foreign students")

```

## Gestión de errores

En caso de error, el comando devuelve Null. Si no se puede acceder al almacén de datos remoto (dirección incorrecta, servidor web no iniciado, http y https no habilitados...), se produce el error 1610 "Ha fallado una petición remota al host XXX". Puede interceptar este error con un método instalado por `ON ERR CALL`.

## `.dataclassName`

► Histórico

`.dataclassName` : 4D.DataClass

### Descripción

Cada clase de datos de un almacén de datos está disponible como una propiedad del objeto `DataStore`. El objeto devuelto contiene una descripción de la clase de datos.

### Ejemplo

```

var $emp : cs.Employee
var $sel : cs.EmployeeSelection
$emp:=ds.Employee // $emp contiene la dataclass Employee
$sel:=$emp.all() // obtiene una selección de entidades de todos los empleados

// también puede escribir directamente:
$sel:=ds.Employee.all()

```

## `.cancelTransaction()`

► Histórico

`.cancelTransaction()` | Parámetros | Tipo | | Descripción | | ----- | ---- | ::| ----- | | | | No requiere ningún parámetro |

### Descripción

La función `.cancelTransaction()` cancela la transacción abierta por la función `.startTransaction()` en el nivel correspondiente del proceso actual para el datastore especificado.

La función `.cancelTransaction()` cancela cualquier cambio realizado en los datos durante la transacción.

Puede anidar varias transacciones (subtransacciones). Si se cancela la transacción principal, también se cancelan todas sus subtransacciones, aunque se hayan validado individualmente mediante la función `.validateTransaction()`.

### Ejemplo

Ver el ejemplo de la función `.startTransaction()`.

## `.clearAllRemoteContexts()`

► Histórico

.clearAllRemoteContexts() | Parámetros | Tipo | Descripción | | ----- | ---- |::| ----- | | | | |  
No requiere ningún parámetro |

## Descripción

The `.clearAllRemoteContexts()` function clears all the attributes for all the active contexts in the datastore.

This function is mainly used in the context of debugging. One thing to keep in mind is that when you open the debugger, it sends requests to the server and queries all the dataclass attributes to display them. This can overload your contexts with unnecessary data.

In such cases, you can use `.clearAllRemoteContexts()` to clear your contexts and keep them clean.

## Ver también

[.getRemoteContextInfo\(\)](#)  
[.getAllRemoteContexts\(\)](#)  
[.setRemoteContextInfo\(\)](#)

## .encryptionStatus()

► Histórico

`.encryptionStatus(): Object`

Parámetros	Tipo		Descripción
Resultado	Objeto	<-	Información sobre el cifrado del almacén de datos actual y de cada tabla

## Descripción

La función `.encryptionStatus()` devuelve un objeto que proporciona el estado de encriptación del archivo de datos actual (es decir, el archivo de datos del `ds` datastore). También se proporciona el estado de cada tabla.

Utilice el comando `Data file encryption status` para determinar el estado de encriptación de cualquier otro archivo de datos.

## Valor devuelto

El objeto devuelto contiene las siguientes propiedades:

Propiedad			Tipo	Descripción
isEncrypted			Booleano	True si el archivo de datos está encriptado
keyProvided			Booleano	True si se proporciona la llave de encriptación que coincide con el archivo de datos encriptados(*) .
tablas			Objeto	Objeto que contiene tantas propiedades como tablas encriptadas o codificadas.
	<i>tableName</i>		Objeto	Tabla encriptada o cifrada
		<i>name</i>	Texto	Nombre de la tabla
		<i>num</i>	Número	Número de tabla
		<i>isEncryptable</i>	Booleano	Verdadero si la tabla está declarada como encriptada en el archivo de estructura
		<i>isEncrypted</i>	Booleano	True si los registros de la tabla están encriptados en el archivo de datos

(\*) Se puede suministrar la llave de encriptación:

- con el comando `.provideDataKey()`,
- en la raíz de un dispositivo conectado antes de abrir el almacén de datos,
- con el comando `Discover data key`.

## Ejemplo

Quiere saber el número de tablas encriptadas en el archivo de datos actual:

```
var $status : Object

$status:=dataStore.encryptionStatus()

If($status.isEncrypted) //la base está encriptada
  C_LONGINT($vcount)
  C_TEXT($tabName)
  For each($tabName;$status.tables)
    If($status.tables[$tabName].isEncrypted)
      $vcount:=$vcount+1
    End if
  End for each
  ALERT(String($vcount)+" encrypted table(s) in this datastore.")
Else
  ALERT("This database is not encrypted.")
End if
```

## .getAllRemoteContexts()

► Histórico

`.getAllRemoteContexts() : Collection`

Parámetros	Tipo		Descripción
Resultado	Objeto	<-	Colección de objetos contextos de optimización

Modo avanzado: esta función está pensada para los desarrolladores que necesitan personalizar las funcionalidades por defecto de ORDA para configuraciones específicas. En la mayoría de los casos, no será necesario utilizarla.

### Descripción

The `.getAllRemoteContexts()` function returns a collection of objects containing information on all the active optimization contexts in the datastore.

For more information on how contexts can be created, see [client/server optimization](#).

Each object in the returned collection has the properties listed in the [`.getRemoteContextInfo\(\)`](#) section.

## Ejemplo

The following code sets up two contexts and retrieves them using `.getAllRemoteContexts()`:

```

var $ds : 4D.DataStoreImplementation
var $persons : cs.PersonsSelection
var $addresses : cs.AddressSelection
var $p : cs.PersonsEntity
var $a : cs.AddressEntity
var $contextA; $contextB : Object
var $info : Collection
var $text : Text

// Open remote datastore
$ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")

// Set context A
$contextA:=New object("context"; "contextA")
$persons:=$ds.Persons.all($contextA)
$text:=""
For each ($p; $persons)
    $text:=$p.firstname+" lives in "+$p.address.city+"
End for each

// Set context B
$contextB:=New object("context"; "contextB")
$addresses:=$ds.Address.all($contextB)
$text:=""
For each ($a; $addresses)
    $text:=$a.zipCode
End for each

// Get all remote contexts (in this case, contextA and contextB)
$info:=$ds.getAllRemoteContexts()
// $info = [{name:"contextB"; dataclass: "Address"; main:"zipCode"}, {name:"contextA"; dataclass:"Persons"; main:"firstname,address.city"}]

```

Este ejemplo sirve como demostración, no está pensado para una implementación real.

## Ver también

[.getRemoteContextInfo\(\)](#)  
[.setRemoteContextInfo\(\)](#)  
[.clearAllRemoteContexts\(\)](#)

## .getInfo()

► Histórico

[.getInfo\(\): Object](#)

Parámetros	Tipo		Descripción
Resultado	Objeto	<-	Propiedades del almacén de datos

### Descripción

La función `.getInfo()` devuelve un objeto que proporciona información sobre el datastore. Esta función es útil para configurar el código genérico.

Objeto devuelto

Propiedad	Tipo	Descripción															
type	cadena	<ul style="list-style-type: none"> <li>"4D": almacén de datos principal, disponible a través de ds</li> <li>"4D Server": almacén de datos remoto, abierto con Open datastore</li> </ul>															
networked	booleano	<ul style="list-style-type: none"> <li>True: el almacén de datos se alcanza a través de una conexión de red.</li> <li>False: no se llega al almacén de datos a través de una conexión de red (base de datos local)</li> </ul>															
localID	texto	ID del almacén de datos en la máquina. Corresponde a la cadena localId dada con el comando <code>Open datastore</code> . Cadena vacía ("") para el almacén de datos principal.															
connection	objeto	<p>Objeto que describe la conexión del almacén de datos remoto (no se devuelve para el almacén de datos principal). Propiedades disponibles:</p> <table border="1"> <thead> <tr> <th>Propiedad</th> <th>Tipo</th> <th>Descripción</th> </tr> </thead> <tbody> <tr> <td>hostname</td> <td>texto</td> <td>Dirección IP o nombre del datastore remoto + ":" + número de puerto</td> </tr> <tr> <td>tls</td> <td>booleano</td> <td>True si se utiliza una conexión segura con el almacén de datos remoto</td> </tr> <tr> <td>idleTimeout</td> <td>number</td> <td>Tiempo de inactividad de la sesión (en minutos)</td> </tr> <tr> <td>user</td> <td>texto</td> <td>Usuario autenticado en el almacén de datos remoto</td> </tr> </tbody> </table>	Propiedad	Tipo	Descripción	hostname	texto	Dirección IP o nombre del datastore remoto + ":" + número de puerto	tls	booleano	True si se utiliza una conexión segura con el almacén de datos remoto	idleTimeout	number	Tiempo de inactividad de la sesión (en minutos)	user	texto	Usuario autenticado en el almacén de datos remoto
Propiedad	Tipo	Descripción															
hostname	texto	Dirección IP o nombre del datastore remoto + ":" + número de puerto															
tls	booleano	True si se utiliza una conexión segura con el almacén de datos remoto															
idleTimeout	number	Tiempo de inactividad de la sesión (en minutos)															
user	texto	Usuario autenticado en el almacén de datos remoto															

- Si la función `.getInfo()` se ejecuta en un 4D Server o 4D monopuesto, `networked` es False.
- Si la función `.getInfo()` se ejecuta en un 4D remoto, `networked` es True

## Ejemplo 1

```
var $info : Object

$info:=ds.getInfo() //Ejecutado en 4D Server o 4D
//>{"type":"4D","networked":false,"localID":""}

$info:=ds.getInfo() // Ejecutado en 4D remoto
//>{"type":"4D","networked":true,"localID":""}
```

## Ejemplo 2

En un almacén de datos remoto:

```
var $remoteDS : cs.DataStore
var $info; $connectTo : Object

$connectTo:=New object("hostname";"111.222.33.44:8044";"user";"marie";"password";"aaaa")
$remoteDS:=Open datastore($connectTo;"students")
$info:=$remoteDS.getInfo()

//{"type":"4D Server",
//"localID":"students",
//"networked":true,
//"connection":{hostname:"111.222.33.44:8044",tls:false,idleTimeout:2880,user:"marie"}}
```

## .getRemoteContextInfo()

`.getRemoteContextInfo(contextName : Text) : Object`

Parámetros	Tipo		Descripción
contextName	Texto	->	Nombre del contexto
Resultado	Objeto	<-	Descripción del contexto

Modo avanzado: esta función está pensada para los desarrolladores que necesitan personalizar las funcionalidades por defecto de ORDA para configuraciones específicas. En la mayoría de los casos, no será necesario utilizarla.

## Descripción

The `.getRemoteContextInfo()` function returns an object that holds information on the `contextName` optimization context in the datastore..

For more information on how optimization contexts can be created, see [client/server optimization](#).

## Objeto devuelto

El objeto devuelto tiene las siguientes propiedades:

Propiedad	Tipo	Descripción
name	Texto	Nombre del contexto
main	Texto	Attribute(s) associated to the context (attribute names are separated by a comma)
dataclass	Texto	Nombre de la clase de datos
currentItem (opcional)	Texto	The attributes of the <a href="#">page mode</a> if the context is linked to a list box. Returned as <code>Null</code> or empty text element if the context name is not used for a list box, or if there is no context for the currentItem

Since contexts behave as filters for attributes, if `main` is returned empty, it means that no filter is applied, and that the server returns all the dataclass attributes.

## Ejemplo

Ver el ejemplo de la sección [.setRemoteContextInfo\(\)](#).

## Ver también

[.setRemoteContextInfo\(\)](#)  
[.getAllRemoteContexts\(\)](#)  
[.clearAllRemoteContexts\(\)](#)

## `.getRequestLog()`

► Histórico

`.getRequestLog() : Collection`

Parámetros	Tipo		Descripción
Resultado	Collection	<-	Colección de objetos, donde cada objeto describe una solicitud

## Descripción

La función `.getRequestLog()` devuelve las peticiones ORDA registradas en memoria del lado del cliente. El registro de peticiones de ORDA debe haberse habilitado previamente mediante la función [.startRequestLog\(\)](#).

Esta función debe ser llamada en un 4D remoto, de lo contrario devuelve una colección vacía. Está diseñado para fines

de depuración en configuraciones cliente/servidor.

#### Valor devuelto

Colección de objetos de petición apilados. La solicitud más reciente tiene el índice 0.

Para una descripción del formato del registro de peticiones de ORDA, consulte la sección [Preguntas del cliente ORDA](#).

#### Ejemplo

Ver el ejemplo 2 de [.startRequestLog\(\)](#).

## .isAdminProtected()

► Histórico

`.isAdminProtected() : Boolean`

Parámetros	Tipo		Descripción
Resultado	Booleano	<-	True si el acceso al Explorador de Datos está desactivado, False si está activado (por defecto)

#### Descripción

La función `.isAdminProtected()` devuelve `True` si el acceso a [Data Explorer](#) ha sido desactivado para la sesión de trabajo.

Por defecto, el acceso al Explorador de Datos se concede para las sesiones `webAdmin`, pero se puede desactivar para evitar cualquier acceso a los datos por parte de los administradores (ver la función [.setAdminProtection\(\)](#)).

#### Ver también

[.setAdminProtection\(\)](#)

## .makeSelectionsAlterable()

► Histórico

`.makeSelectionsAlterable()` | Parámetros | Tipo | | Descripción | | ----- | ---- | ::| ----- | | | | | No requiere ningún parámetro |

#### Descripción

La función `.makeSelectionsAlterable()` establece todas las selecciones de entidades como alterables por defecto en los datastores de la aplicación actual (incluyendo [datastores remotos](#)). Está pensado para ser utilizado una vez, por ejemplo en el método base `On Startup`.

Cuando no se llama a esta función, las nuevas selecciones de entidades pueden ser compatibles, dependiendo de la naturaleza de su "padre", o de [cómo se crean](#).

Esta función no modifica las selecciones de entidades creadas por `.copy()` o `OB Copy` cuando se utiliza la opción explícita `ck shared`.

Compatibilidad: esta función sólo debe utilizarse en proyectos convertidos desde versiones de 4D anteriores a 4D v18 R5 y que contengan llamadas `.add()`. En este contexto, el uso de `.makeSelectionsAlterable()` puede ahorrar tiempo al restaurar instantáneamente el comportamiento anterior de 4D en los proyectos existentes. Por otro lado, utilizar este método en proyectos nuevos creados en 4D v18 R5 y superiores no es recomendable, ya que impide compartir las selecciones de entidades, lo que ofrece mayor rendimiento y escalabilidad.

## .provideDataKey()

► Histórico

.provideDataKey( *curPassPhrase* : Text ) : Object

.provideDataKey( *curDataKey* : Object ) : Object

Parámetros	Tipo		Descripción
<i>curPassPhrase</i>	Texto	->	Frase de cifrado actual
<i>curDataKey</i>	Objeto	->	Llave de encriptación de datos actual
Resultado	Objeto	<-	Resultado de la coincidencia de la llave de encriptación

### Descripción

La función `.provideDataKey()` permite suministrar una llave de cifrado de datos para el archivo de datos actual del datastore y detecta si la llave coincide con los datos cifrados. Esta función se puede utilizar al abrir una base encriptada, o al ejecutar cualquier operación de encriptación que requiera la llave de encriptación, como por ejemplo volver a encriptar el archivo de datos.

- La función `.provideDataKey()` debe ser llamada en una base de datos encriptada. Si se llama en una base no cifrada, se devuelve el error 2003 (la llave de cifrado no coincide con los datos.). Utilice el comando `Estado de cifrado del archivo de datos` para determinar si la base de datos está cifrada.
- La función `.provideDataKey()` no puede ser llamada desde un 4D remoto o un datastore remoto encriptado.

Si utiliza el parámetro *curPassPhrase*, pase la cadena utilizada para generar la llave de cifrado de datos. Cuando se utiliza este parámetro, se genera una llave de encriptación.

Si utiliza el parámetro *curDataKey*, pase un objeto (con la propiedad `encodedKey`) que contenga la llave de cifrado de los datos. Esta llave puede haber sido generada con el comando `New data key`.

Si se aporta una llave de cifrado de datos válida, se añade a la `keyChain` de la memoria y se activa el modo de cifrado:

- todas las modificaciones de datos en las tablas encriptadas se cifran en el disco (.4DD, .journal, 4Dindx)
- todos los datos cargados desde tablas encriptadas se descifran en memoria

### Resultado

El resultado de la orden se describe en el objeto devuelto:

Propiedad		Tipo	Descripción
<i>success</i>		Booleano	True si la llave de encriptación proporcionada coincide con los datos encriptados, False en caso contrario
			Las siguientes propiedades se devuelven sólo si <i>success</i> es FALSE
<i>status</i>		Número	Código de error (4 si la llave de encriptación suministrada es errónea)
<i>statusText</i>		Texto	Mensaje de error
<i>errors</i>		Collection	Pila de errores. El primer error tiene el índice más alto
	[ ].componentSignature	Texto	Nombre del componente interno
	[ ].errCode	Número	Número de error
	[ ].message	Texto	Mensaje de error

Si no se da *curPassphrase* o *curDataKey*, `.provideDataKey()` devuelve null (no se genera ningún error).

## Ejemplo

```
var $keyStatus : Object
var $passphrase : Text

$passphrase:=Request("Enter the passphrase")
If(OK=1)
    $keyStatus:=ds.provideDataKey($passphrase)
    If($keyStatus.success)
        ALERT("You have provided a valid encryption key")
    Else
        ALERT("You have provided an invalid encryption key, you will not be able to work with encrypted d
End if
End if
```

## .setAdminProtection()

► Histórico

.setAdminProtection( *status* : Boolean )

Parámetros	Tipo		Descripción
status	Booleano	- >	True para desactivar el acceso Data Explorer a los datos del puerto <code>webAdmin</code> , False (por defecto) para otorgar el acceso

### Descripción

La función `.setAdminProtection()` permite deshabilitar cualquier acceso a datos en el [puerto de administración web](#), incluso para el [Explorador de datos](#) en las sesiones de `WebAdmin`.

Por defecto, cuando no se llama a la función, el acceso a los datos se concede siempre en el puerto de administración web para una sesión con privilegio `WebAdmin` utilizando el Explorador de Datos. En algunas configuraciones, por ejemplo, cuando el servidor de aplicaciones está alojado en una máquina de terceros, es posible que no desee que el administrador pueda ver sus datos, aunque puede editar la configuración del servidor, incluyendo la configuración de la [access key](#).

En este caso, puede llamar a esta función para deshabilitar el acceso a los datos del Explorador de Datos en el puerto de administración web de la máquina, incluso si la sesión de usuario tiene el privilegio `WebAdmin`. Cuando se ejecuta esta función, el archivo de datos se protege inmediatamente y el estado se almacena en el disco: el archivo de datos estará protegido incluso si se reinicia la aplicación.

## Ejemplo

Se crea un método proyecto `protectDataFile` para llamar antes de los despliegues, por ejemplo:

```
ds.setAdminProtection(True) //Desactiva el acceso a los datos del Explorador de Datos
```

### Ver también

[.isAdminProtected\(\)](#)

## .setRemoteContextInfo()

► Histórico

.setRemoteContextInfo( *contextName* : Text ; *dataClassName* : Text ; *attributes* : Text {; *contextType* : Text { ; *pageLength* : Integer}}})  
.setRemoteContextInfo( *contextName* : Text ; *dataClassName* : Text; *attributesColl* : Collection {; *contextType* : Text {

```

; pageLength : Integer }} )  

.setRemoteContextInfo( contextName : Text ; dataClassObject : 4D.DataClass ; attributes : Text {; contextType : Text {  

; pageLength : Integer }})  

.setRemoteContextInfo( contextName : Text ; dataClassObject : 4D.DataClass ; attributesColl : Collection {;  

contextType : Text { ; pageLength : Integer }} )

```

Parámetros	Tipo		Descripción
contextName	Texto	->	Nombre del contexto
dataClassName	Texto	->	Nombre de la dataclass
dataClassObject	4D.DataClass	->	dataclass object (e.g datastore. Employee)
attributes	Texto	->	Lista de atributos separados por comas
attributesColl	Collection	->	Colección de nombres de atributos (text)
contextType	Texto	->	If provided, value must be "main" or "currentItem"
pageLength	Integer	->	Page length of the entity selection linked to the context (default is 80)

Modo avanzado: esta función está pensada para los desarrolladores que necesitan personalizar las funcionalidades por defecto de ORDA para configuraciones específicas. En la mayoría de los casos, no será necesario utilizarla.

## Descripción

The `.setRemoteContextInfo()` function links the specified dataclass attributes to the `contextName` optimization context. If an optimization context already exists for the specified attributes, this command replaces it.

When you pass a context to the ORDA class functions, the REST request optimization is triggered immediately:

- the first entity is not fully loaded as done in automatic mode
- pages of 80 entities (or `pageLength` entities) are immediately asked to the server with only the attributes in the context

For more information on how optimization contexts are built, refer to the [client/server optimization paragraph](#)

In `contextName`, pass the name of the optimization context to link to the dataclass attributes.

To designate the dataclass that will receive the context, you can pass a `dataClassName` or a `dataClassObject`.

To designate the attributes to link to the context, pass either a list of attributes separated by a comma in `attributes` (Text), or a collection of attribute names in `attributesColl` (collection of text).

If `attributes` is an empty Text, or `attributesColl` is an empty collection, all the scalar attributes of the dataclass are put in the optimization context. If you pass an attribute that does not exist in the dataclass, the function ignores it and an error is thrown.

You can pass a `contextType` to specify if the context is a standard context or the context of the current entity selection item displayed in a list box:

- If set to "main" (default), the `contextName` designates a standard context.
- If set to "currentItem", the attributes passed are put in the context of the current item. See [Entity selection-based list box](#).

In `pageLength`, specify the number of dataclass entities to request from the server.

You can pass a `pageLength` for a relation attribute which is an entity selection (one to many). The syntax is `relationAttributeName:pageLength` (e.g employees:20).

## Ejemplo 1

```

var $ds : 4D.DataStoreImplementation
var $persons : cs.PersonsSelection
var $p : cs.PersonsEntity
var $contextA : Object
var $info : Object
var $text : Text

// Open remote datastore
$ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")

// Set context info
$contextA:=New object("context"; "contextA")
$ds.setRemoteContextInfo("contextA"; $ds.Persons; "firstname, lastname")

// Send requests to the server using a loop
$persons:=$ds.Persons.all($contextA)
$text:=""
For each ($p; $persons)
    $text:=$p.firstname + " " + $p.lastname
End for each

// Check contents of the context
$info:=$ds.getRemoteContextInfo("contextA")
// $info = {name:"contextA";dataclass:"Persons";main:"firstname, lastname"}

```

Este ejemplo sirve como demostración, no está pensado para una implementación real.

## Ejemplo 2

The following piece of code requests pages of 30 entities of the `Address` dataclass from the server. The returned entities only contain the `zipCode` attribute.

For each `Address` entity, 20 Persons entities are returned, and they only contain the `lastname` and `firstname` attributes:

```

var $ds : 4D.DataStoreImplementation

$ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")

$ds.setRemoteContextInfo("contextA"; $ds.Address; "zipCode, persons:20,\n
persons.lastname, persons.firstname"; "main"; 30)

```

## Ejemplo 3 - Listbox

```

// When the form loads
Case of
  : (Form event code=On Load)

    Form.ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")

    // Set the attributes of the page context
    Form.ds.setRemoteContextInfo("LB"; Form.ds.Persons; "age, gender,\n
      children"; "currentItem")

    Form.settings:=New object("context"; "LB")
    Form.persons:=Form.ds.Persons.all(Form.settings)
    // Form.persons is displayed in a list box
End case

// When you get the attributes in the context of the current item: Form.currentItemLearntAttributes:=For
// Form.currentItemLearntAttributes = "age, gender, children"

```

Ver también

[.getRemoteContextInfo\(\)](#)  
[.getAllRemoteContexts\(\)](#)  
[.clearAllRemoteContexts\(\)](#)

## .startRequestLog()

► Histórico

[.startRequestLog\(\)](#)  
[.startRequestLog\( file : 4D.File \)](#)  
[.startRequestLog\( reqNum : Integer \)](#)

Parámetros	Tipo		Descripción
file	4D.File	->	Objeto File
reqNum	Integer	->	Número de peticiones a mantener en memoria

### Descripción

La función `.startRequestLog()` inicia el registro de las peticiones ORDA del lado del cliente.

Esta función debe ser llamada en un 4D remoto, de lo contrario no hace nada. Está diseñado para fines de depuración en configuraciones cliente/servidor.

El registro de peticiones ORDA puede ser enviado a un archivo o a la memoria, dependiendo del tipo de parámetro:

- Si se pasa un objeto `file` creado con el comando `File`, los datos de registro se escriben en este archivo como una colección de objetos (formato JSON). Cada objeto representa una petición.  
 Si el archivo no existe ya, se crea. En caso contrario, si el archivo ya existe, los nuevos datos de registro se añaden a él. Si se llama a `.startRequestLog( )` con un archivo mientras se inició previamente un registro en memoria, el registro en memoria se detiene y se vacía.

Debe añadirse manualmente un carácter `¥N` al final del archivo para realizar una validación JSON

- Si se pasa un entero `reqNum`, se vacía el registro en memoria (si lo hay) y se inicializa un nuevo registro. Mantendrá `reqNum` peticiones en memoria hasta que se alcance el número, en cuyo caso se vacían las entradas más antiguas (pila FIFO).  
 Si se llama a `.startRequestLog()` con un `reqNum` mientras se ha iniciado previamente un registro en un archivo, se detiene el registro del archivo.

- Si no ha pasado ningún parámetro, el registro se inicia en la memoria. Si se llamó previamente a `.startRequestLog()` con un `reqNum` (antes de un `.stopRequestLog()`), los datos del registro se apilan en memoria hasta la próxima vez que se vacíe el registro o se llame a `.stopRequestLog()`.

Para una descripción del formato del registro de peticiones de ORDA, consulte la sección [Preguntas del cliente ORDA](#).

## Ejemplo 1

Desea registrar las solicitudes de los clientes ORDA en un archivo y utilizar el número de secuencia del registro:

```
var $file : 4D.File
var $e : cs.PersonsEntity

$file:=File("/LOGS/ORDARequests.txt") //logs folder

SET DATABASE PARAMETER(Client Log Recording;1) //to trigger the global log sequence number
ds.startRequestLog($file)
$e:=ds.Persons.get(30001) //send a request
ds.stopRequestLog()
SET DATABASE PARAMETER(Client Log Recording;0)
```

## Ejemplo 2

Quiere registrar las peticiones de los clientes ORDA en la memoria:

```
var $es : cs.PersonsSelection
var $log : Collection

ds.startRequestLog(3) //keep 3 requests in memory

$es:=ds.Persons.query("name=:1";"Marie")
$es:=ds.Persons.query("name IN :1";New collection("Marie"))
$es:=ds.Persons.query("name=:1";"So@")

$log:=ds.getRequestLog()
ALERT("The longest request lasted: "+String($log.max("duration"))+" ms")
```

## .startTransaction()

► Histórico

`.startTransaction()` | Parámetros | Tipo | | Descripción | | ----- | ---- | | ----- | | | | No requiere ningún parámetro |

### Descripción

La función `.startTransaction()` inicia una transacción en el proceso actual en la base de datos que coincide con el datastore al que se aplica. Todos los cambios realizados en las entidades del almacén de datos en el proceso de la transacción se almacenan temporalmente hasta que la transacción se valida o se cancela.

Si se llama a este método en el almacén de datos principal (es decir, el almacén de datos devuelto por el comando `ds`), la transacción se aplica a todas las operaciones realizadas en el almacén de datos principal y en la base de datos subyacente, incluyendo por tanto ORDA y los lenguajes clásicos.

Puede anidar varias transacciones (subtransacciones). Cada transacción o sub-transacción debe ser eventualmente cancelada o validada. Note que si se cancela la transacción principal, también se cancelan todas sus subtransacciones, aunque se hayan validado individualmente mediante la función `.validateTransaction()`.

## Ejemplo

```

var $connect; $status : Object
var $person : cs.PersonsEntity
var $ds : cs.DataStore
var $choice : Text
var $error : Boolean

Case of
  :($choice=="local")
    $ds:=ds
  :($choice=="remote")
    $connect:=New object("hostname";"111.222.3.4:8044")
    $ds:=Open datastore($connect;"myRemoteDS")
End case

$ds.startTransaction()
$person:=$ds.Persons.query("lastname=:1";"Peters").first()

If($person#Null)
  $person.lastname:="Smith"
  $status:=$person.save()
End if
...
...
If($error)
  $ds.cancelTransaction()
Else
  $ds.validateTransaction()
End if

```

## .stopRequestLog()

► Histórico

.stopRequestLog()  
 | Parámetros | Tipo | Descripción | | ----- | ---- | | ----- | | | | No requiere ningún parámetro |

### Descripción

La función `.stopRequestLog()` detiene cualquier registro de peticiones ORDA del lado del cliente (en archivo o en memoria). Es particularmente útil cuando se registra un archivo, ya que realmente cierra el documento abierto en el disco.

Esta función debe ser llamada en un 4D remoto, de lo contrario no hace nada. Está diseñado para fines de depuración en configuraciones cliente/servidor.

### Ejemplo

Ver ejemplos de [.startRequestLog\(\)](#).

## .validateTransaction()

► Histórico

.validateTransaction()  
 | Parámetros | Tipo | Descripción | | ----- | ---- | | ----- | | | | No requiere ningún parámetro |

### Descripción

La función `.validateTransaction()` acepta la transacción que se inició con [.startTransaction\(\)](#) en el nivel correspondiente del datastore especificado.

La función guarda los cambios en los datos del almacén de datos que se produjeron durante la transacción.

Puede anidar varias transacciones (subtransacciones). Si se cancela la transacción principal, también se cancelan todas sus subtransacciones, aunque se hayan validado individualmente utilizando esta función.

## Ejemplo

Ver el ejemplo de [.startTransaction\(\)](#).

# Email

La creación, el envío o la recepción de correos electrónicos en 4D se realiza manejando un objeto `Email`.

Objetos `Email` se crean cuando se reciben correos a través de una función de clase `transporter`:

- IMAP - funciones `.getMail()` y `.getMails()` para obtener los correos electrónicos de un servidor IMAP
- POP3 - función `.getMail()` para obtener un correo electrónico de un servidor POP3.

También puede crear un nuevo objeto `Email` en blanco llamando al comando 4D `New object`, y luego llenarlo con [propiedades del objeto de correo electrónico](#).

Los objetos `Email` se envían utilizando la función SMTP `.send()`.

Los comandos `MAIL Convert from MIME` y `MAIL Convert to MIME` pueden utilizarse para convertir objetos `Email` a y desde contenidos MIME.

## Objeto Email

Los objetos Email ofrecen las siguientes propiedades:

4D sigue la especificación [JMAP](#) para formatear el objeto Email.

### `.attachments : Collection`

colección de objeto(s) `4D.MailAttachment`

### `.bcc : Text`

### `.bcc : Object`

### `.bcc : Collection`

copia carbón invisible (BCC) de las [dirección\(es\)](#) de los destinatarios del correo electrónico

### `.bodyStructure : Object`

`EmailBodyPart`, es decir, la estructura MIME completa del cuerpo del mensaje (opcional)

### `.bodyValues : Object`

`EmailBodyValue`, que contiene un objeto para cada `<partID>` de `bodyStructure` (opcional)

### `.cc : Text`

### `.cc : Object`

### `.cc : Collection`

Carbon Copy (CC) adicional del destinatario del correo electrónico [addresse\(s\)](#) del correo electrónico

### `.comments : Text`

encabezado de comentarios adicionales

### `.from : Text`

### `.from : Object`

### `.from : Collection`

Dirección(es) de origen del correo electrónico

### `.headers : Collection`

colección de objetos `EmailHeader`, en el orden en que aparecen en el mensaje

### `.htmlBody : Text`

	representación HTML del mensaje de correo electrónico (el conjunto de caracteres por defecto es UTF-8) (opcional, sólo SMTP)
.id : Text	id único del servidor IMAP
.inReplyTo : Text	identificador(es) del mensaje(s) original(es) al que el mensaje actual es una respuesta
.keywords : Object	conjunto de palabras clave como objeto, donde cada nombre de propiedad es una palabra clave y cada valor es verdadero
.messageId : Text	encabezado de identificador de mensaje ("message-id")
.receivedAt : Text	fecha de llegada del correo electrónico al servidor IMAP en formato ISO 8601 UTC (por ejemplo: 2020-09-13T16:11:53Z)
.references: Collection	Colección de todos los identificadores de mensajes de la cadena de respuestas anterior
.replyTo : Text	
.replyTo : Object	
.replyTo : Collection	
	dirección(es) para las respuestas
.sendAt : Text	sello de tiempo del correo en formato ISO 8601 UTC
.sender : Text	
.sender : Object	
.sender : Collection	
	addresse(s) source(s) del correo electrónico
.size : Integer	tamaño (expresado en bytes) del objeto Email devuelto por el servidor IMAP
.subject : Text	descripción del tema
.textBody : Text	Representación en texto plano del mensaje de correo electrónico (el conjunto de caracteres por defecto es UTF-8) (opcional, sólo SMTP)
.to : Text	
.to : Object	
.to : Collection	
	destinatario principal dirección(es) del correo electrónico

## Direcciones de correo electrónico

Todas las propiedades que contienen direcciones de correo electrónico (`from`, `cc`, `bcc`, `to`, `sender`, `replyTo`) aceptan un valor de tipo texto, objeto o colección.

Texto

- correo electrónico único: "somebody@domain.com"
- un nombre+email: "Somebody somebody@domain.com"
- combinación de varios emails: "Somebody somebody@domain.com,me@home.org"

## Objeto

Un objeto con dos propiedades:

Propiedad	Tipo	Descripción
name	Texto	Nombre a mostrar (puede ser null)
email	Texto	Correo electrónico

## Collection

Una colección de objetos dirección.

## Gestión del cuerpo del correo electrónico

Las propiedades `textBody` y `htmlBody` sólo se utilizan con la función `SMTP.send()` para permitir el envío de correos sencillos. Cuando se llenan ambas propiedades, se utiliza el tipo MIME content-type multipart/alternative. El cliente de correo electrónico debería entonces reconocer la parte multipart/alternative y mostrar la parte texto o html según sea necesario.

`bodyStructure` y `bodyValues` se utilizan para `SMTP` cuando el `objeto Email` se construye a partir de un documento MIME, por ejemplo, cuando se genera mediante el comando `MAIL Convert from MIME`. En este caso, las propiedades `bodyStructure` y `bodyValues` deben pasarse juntas, y no se recomienda utilizar `textBody` y `htmlBody`.

### Ejemplo de objetos bodyStructure y bodyValues

```
"bodyStructure": {
  "type": "multipart/mixed",
  "subParts": [
    {
      "partId": "p0001",
      "type": "text/plain"
    },
    {
      "partId": "p0002",
      "type": "text/html"
    }
  ],
  "bodyValues": {
    "p0001": {
      "value": "I have the most brilliant plan. Let me tell you all about it."
    },
    "p0002": {
      "value": "<!DOCTYPE html><html><head><title></title><style type=\"text/css\">div{font-size:16px}</st
    }
  }
}
```

## .attachments

`.attachments : Collection`

Descripción

La propiedad `.attachments` contiene una colección de objeto(s) `4D.MailAttachment`.

Los objetos Attachment (adjuntos) se definen mediante el comando `MAIL New attachment`. Los objetos adjuntos tienen [propiedades y funciones](#) específicas.

## .bcc

`.bcc` : Text  
`.bcc` : Object  
`.bcc` : Collection

### Descripción

La propiedad `.bcc` contiene la copia carbón invisible (BCC) de las [dirección\(es\)](#) de los destinatarios del correo electrónico.

## .bodyStructure

`.bodyStructure` : Object

### Descripción

La propiedad `.bodyStructure` contiene el objeto `EmailBodyPart`, es decir, la estructura MIME completa del cuerpo del mensaje (opcional). Ver la sección [Gestión del cuerpo](#).

El objeto `.bodyStructure` contiene las siguientes propiedades:

Propiedad	Tipo	Valor
partID	Texto	Identifica la parte de manera única dentro del correo electrónico
type	Texto	(obligatorio) Valor del campo del encabezado Content-Type de la parte
charset	Texto	Valor del parámetro charset del campo del encabezado Content-Type
encoding	Texto	Si <code>isEncodingProblem=true</code> , se añade el valor Content-Transfer-Encoding (por defecto indefinido)
disposition	Texto	Valor del campo del encabezado Content-Disposition de la parte
lenguaje	Colección de textos	Lista de etiquetas de lenguaje, como se define en la <a href="#">RFC3282</a> , en el campo del encabezado Content-Language de la parte, si está presente.
location	Texto	URI, como se define en la <a href="#">RFC2557</a> , en el campo del encabezado Content-Location de la parte, si está presente.
subParts	Colección de objetos	Partes del cuerpo de cada hijo (colección de objetos <code>EmailBodyPart</code> )
headers	Colección de objetos	Lista de todos los campos del encabezado de la parte, en el orden en que aparecen en el mensaje (colección de objetos <code>EmailHeader</code> , ver la propiedad <code>headers</code> )

## .bodyValues

`.bodyValues` : Object

### Descripción

La propiedad `.bodyValues` contiene el objeto `EmailBodyValue`, que contiene un objeto para cada `<partID>` de `bodyStructure` (opcional). Ver la sección [Gestión del cuerpo](#).

El objeto `.bodyValues` contiene las siguientes propiedades:

Propiedad	Tipo	Valor
<code>partID.value</code>	texto	Valor de la parte del cuerpo
<code>partID.isEncodingProblem</code>	booleano	True si se encuentran secciones malformadas al decodificar el conjunto de caracteres, o el conjunto de caracteres desconocido, o la codificación de transferencia de contenido desconocida. Falso por defecto

## .CC

`.cc` : Text  
`.cc` : Object  
`.cc` : Collection

### Descripción

La propiedad `.cc` contiene la Carbon Copy (CC) adicional del destinatario del correo electrónico [addresse\(s\)](#) del correo electrónico.

## .comments

`.comments` : Text

### Descripción

La propiedad `.comments` contiene un encabezado de comentarios adicionales.

Los comentarios sólo aparecen en la sección del encabezado del mensaje (manteniendo el cuerpo del mensaje intacto).

Para conocer los requisitos específicos de formato, consulte la [RFC#5322](#).

## .from

`.from` : Text  
`.from` : Object  
`.from` : Collection

### Descripción

La propiedad `.from` contiene las [Dirección\(es\)](#) de origen del correo electrónico.

Cada email que se envía tiene las direcciones `sender` y `from`:

- el dominio `sender` es el que obtiene el servidor de recepción del email al abrir la sesión,
- la dirección `from` es lo que verán los destinatarios.

Para mejorar la entregabilidad, se recomienda utilizar las mismas direcciones para `from` y `sender`.

## .headers

`.headers` : Collection

### Descripción

La propiedad `.headers` contiene una colección de objetos `EmailHeader`, en el orden en que aparecen en el mensaje. Esta propiedad permite a los usuarios añadir encabezados extendidos (registrados) o definidos por el usuario (no registrados, que comienzan por "X").

Si una propiedad del objeto `EmailHeader` define un encabezado como "from" o "cc" que ya está definido como una propiedad a nivel de correo, la propiedad `EmailHeader` se ignora.

Cada objeto de la colección de encabezados puede contener las siguientes propiedades:

Propiedad	Tipo	Valor
<code>[].name</code>	texto	(obligatorio) Nombre del campo de encabezado según se define en <a href="#">RFC#5322</a> . Si es null o indefinido, el campo encabezado no se agrega al encabezado MIME.
<code>[].value</code>	texto	Valores de los campos encabezado definidos en <a href="#">RFC#5322</a>

## .htmlBody

`.htmlBody` : Text

Descripción

La propiedad `.htmlBody` contiene la representación HTML del mensaje de correo electrónico (el conjunto de caracteres por defecto es UTF-8) (opcional, sólo SMTP). Ver la sección [Gestión del cuerpo](#).

## .id

`.id` : Text

Descripción

[IMAP transporter](#) únicamente.

La propiedad `.id` contiene el id único del servidor IMAP.

## .inReplyTo

`.inReplyTo` : Text

Descripción

La propiedad `.inReplyTo` contiene el (los) identificador(es) del mensaje(s) original(es) al que el mensaje actual es una respuesta.

Para conocer los requisitos específicos de formato, consulte la [RFC#5322](#).

## .keywords

`.keywords` : Object

Descripción

La propiedad `.keywords` contiene un conjunto de palabras clave como objeto, donde cada nombre de propiedad es una palabra clave y cada valor es verdadero.

Esta propiedad es el encabezado "keywords" (ver [RFC#4021](#)).

Propiedad	Tipo	Valor
<code>. &lt;keyword&gt;</code>	booleano	Palabra clave a definir (el valor debe ser true)

Palabras clave reservadas:

- `$draft` - Indica que un mensaje es un borrador
- `$seen` - Indica que se ha leído un mensaje
- `$flagged` - Indica que un mensaje necesita atención especial (por ejemplo, urgente)
- `$answered` - Indica que se ha respondido un mensaje
- `$deleted` - Indica un mensaje a eliminar

## Ejemplo

```
$mail.keywords["$flagged"] := True
$mail.keywords["4d"] := True
```

## .messageId

`.messageId` : Text

### Descripción

La propiedad `.messageId` contiene un encabezado de identificador de mensaje ("message-id").

Este encabezado suele ser "lettersOrNumbers@domainname", por ejemplo, "[abcdef.123456@4d.com](mailto:abcdef.123456@4d.com)". Este identificador único se utiliza, en particular, en foros o listas de correo públicas. En general, los servidores de correo añaden automáticamente este encabezado a los mensajes que envían.

## .receivedAt

`.receivedAt` : Text

### Descripción

[IMAP transporter](#) únicamente.

La propiedad `.receivedAt` contiene la fecha de llegada del correo electrónico al servidor IMAP en formato ISO 8601 UTC (por ejemplo: 2020-09-13T16:11:53Z).

## .references

`.references`: Collection

### Descripción

La propiedad `.references` contiene la Colección de todos los identificadores de mensajes de la cadena de respuestas anterior.

Para conocer los requisitos específicos de formato, consulte la [RFC#5322](#).

## .replyTo

`.replyTo` : Text  
`.replyTo` : Object  
`.replyTo` : Collection

### Descripción

La propiedad `.replyTo` contiene la dirección(es) para las respuestas.

## .sendAt

.sendAt : Text

#### Descripción

La propiedad `.sendAt` contiene el sello de tiempo del correo en formato ISO 8601 UTC.

### .sender

.sender : Text

.sender : Object

.sender : Collection

#### Descripción

La propiedad `.sender` contiene las [addresse\(s\) source\(s\)](#) del correo electrónico.

Cada email que se envía tiene las direcciones sender y [from](#):

- el dominio sender es el que obtiene el servidor de recepción del email al abrir la sesión,
- la dirección from es lo que verán los destinatarios.

Para mejorar la entregabilidad, se recomienda utilizar las mismas direcciones para from y sender.

### .size

.size : Integer

#### Descripción

[IMAP transporter](#) únicamente.

La propiedad `.size` contiene el tamaño (expresado en bytes) del objeto Email devuelto por el servidor IMAP.

### .subject

.subject : Text

#### Descripción

La propiedad `.subject` contiene la descripción del tema.

### .textBody

.textBody : Text

#### Descripción

La propiedad `.textBody` contiene la Representación en texto plano del mensaje de correo electrónico (el conjunto de caracteres por defecto es UTF-8) (opcional, sólo SMTP). Ver la sección [Gestión del cuerpo](#).

### .to

.to : Text

.to : Object

.to : Collection

#### Descripción

La propiedad `.to` contiene el destinatario principal [dirección\(es\)](#) del correo electrónico.

# MAIL Convert from MIME

► Histórico

MAIL Convert from MIME( *mime* : Blob ) : Object

MAIL Convert from MIME( *mime* : Text ) : Object

Parámetros	Tipo		Descripción
<i>mime</i>	Blob, Text	->	Email en MIME
Resultado	Objeto	<-	Objeto Email

## Descripción

El comando `MAIL Convert from MIME` convierte un documento MIME en un objeto de correo electrónico válido.

4D sigue la especificación [JMAP](#) para formatear el objeto email devuelto.

Pase en *mime* un documento MIME válido para convertir. Puede ser suministrado por cualquier servidor o aplicación de correo. Puede pasar un BLOB o un texto en el parámetro *mime*. Si el MIME proviene de un archivo, se recomienda utilizar un parámetro BLOB para evitar problemas relacionados con las conversiones del conjunto de caracteres y los saltos de línea.

## Objeto devuelto

Objeto Email.

## Ejemplo 1

Quiere cargar una plantilla de correo guardada como MIME en un documento de texto y enviar un correo electrónico:

```
var $mime: Blob
var $mail;$server;$transporter;$status: Object

$mime:=File("/PACKAGE/Mails/templateMail.txt").getContent()

$mail:=MAIL Convert from MIME($mime)
$mail.to:="smith@mail.com"
$mail.subject:="Hello world"

$server:=New object
$server.host:="smtp.gmail.com"
$server.port:=465
$server.user:="test@gmail.com"
$server.password:="XXXX"

$transporter:=SMTP New transporter($server)
$status:=$transporter.send($mail)
```

## Ejemplo 2

En este ejemplo, se envía directamente un documento de 4D Write Pro que contiene imágenes:

```

var $mime: Blob
var $email;$server;$transporter;$status: Object

// Exportación Mime del documento 4D Write Pro
WP EXPORT VARIABLE(WParea;$mime;wk mime html)

// convertir la variable Mime de 4D Write Pro en objeto email
$email:=MAIL Convert from MIME($mime)

// Llenar los encabezados del objeto email
$email.subject:="4D Write Pro HTML body"
$email.from:="YourEmail@gmail.com"
$email.to:="RecipientEmail@mail.com"

$server:=New object
$server.host:="smtp.gmail.com"
$server.port:=465
$server.user:="YourEmail@gmail.com"
$server.password:="XXXX"

$transporter:=SMTP New transporter($server)
$status:=$transporter.send($email)

```

## MAIL Convert to MIME

► Histórico

**MAIL Convert to MIME( *mail* : Object { ; *options* : Object } ) : Text**

Parámetros	Tipo		Descripción
mail	Objeto	->	Objeto Email
options	Objeto	->	Opciones de codificación y de charset del mail
Resultado	Texto	<-	Objeto email convertido en MIME

### Descripción

El comando `MAIL Convert to MIME` convierte un objeto de correo electrónico en texto MIME. Este comando es llamado internamente por `SMTP_transporter.send( )` para formatear el objeto de correo electrónico antes de enviarlo. Se puede utilizar para analizar el formato MIME del objeto.

En *mail*, pase el contenido y los detalles de la estructura del correo electrónico a convertir. Esto incluye información como las direcciones de correo electrónico (remitente y destinatario(s)), el propio mensaje y el tipo de visualización del mensaje.

4D sigue la especificación [JMAP](#) para formatear el objeto email.

En *options*, puede configurar la codificación y el charset del mail. Las siguientes propiedades están disponibles:

Propiedad	Tipo	Descripción		
headerCharset	Texto	Charset y codificación utilizados para las siguientes partes del correo electrónico: asunto, nombres de archivos adjuntos y atributo(s) del nombre del correo electrónico. Valores posibles:		
		Constante	Valor	Comentario
		mail mode ISO2022JP	US-ASCII_ISO-2022-JP_UTF8_QP	<ul style="list-style-type: none"> <li>• <i>headerCharset</i>: US-ASCII si es posible, japonés (ISO-2022-JP) &amp; Quoted-printable si es posible, de lo contrario UTF-8 &amp; Quoted-printable</li> <li>• <i>bodyCharset</i>: US-ASCII if possible, Japanese (ISO-2022-JP) &amp; 7-bit if possible, otherwise UTF-8 &amp; Quoted-printable</li> </ul>
		mail mode ISO88591	ISO-8859-1	<ul style="list-style-type: none"> <li>• <i>headerCharset</i>: ISO-8859-1 &amp; Quoted-printable</li> <li>• <i>bodyCharset</i>: ISO-8859-1 &amp; 8-bit</li> </ul>
		mail mode UTF8	US-ASCII_UTF8_QP	<i>headerCharset</i> & <i>bodyCharset</i> : US-ASCII si es posible, de lo contrario UTF-8 & Quoted-printable (valor por defecto)
		mail mode UTF8 in base64	US-ASCII_UTF8_B64	<i>headerCharset</i> & <i>bodyCharset</i> : US-ASCII si es posible, de lo contrario UTF-8 & base64
bodyCharset	Texto	Charset y codificación utilizados para el contenido html y el texto del cuerpo del correo electrónico. Valores posibles: los mismos que para headerCharset (ver arriba)		

Si se omite el parámetro *options*, se utiliza la configuración del modo de correo UTF8 para las partes encabezado y cuerpo.

### Ejemplo

```
var $mail: Object
var $mime: Text
$mail:=New object

// Creación de un correo
$mail.from:="tsales@massmarket.com"
$mail.subject:="Terrific Sale! This week only!"
$mail.textBody:="Text format email"
$mail.htmlBody:="<html><body>HTML format email</body></html>"
$mail.to:=New collection
$mail.to.push(New object ("email";"noreply@4d.com"))
$mail.to.push(New object ("email";"test@4d.com"))

// transforma el objeto mail en MIME
$mime:=MAIL Convert to MIME($mail)

// Contenidos de $mime:
// MIME-Version: 1.0
// Date: Thu, 11 Oct 2018 15:42:25 GMT
// Message-ID: <7CA5D25B2B5E0047A36F2E8CB30362E2>
// Sender: tsales@massmarket.com
// From: tsales@massmarket.com
// To: noreply@4d.com
// To: test@4d.com
// Content-Type: multipart/alternative; boundary="E0AE5773D5E95245BBBBD80DD0687E218"
// Subject: Terrific Sale! This week only!
//
// --E0AE5773D5E95245BBBBD80DD0687E218
// Content-Type: text/plain; charset="UTF-8"
// Content-Transfer-Encoding: quoted-printable
//
// Text format email
// --E0AE5773D5E95245BBBBD80DD0687E218
// Content-Type: text/html; charset="UTF-8"
// Content-Transfer-Encoding: quoted-printable
//
// <html><body>HTML format email</body></html>
// --E0AE5773D5E95245BBBBD80DD0687E218--
```

# Entity

Una [entidad](#) es una instancia de una [Dataclass](#), como un registro de la tabla que coincide con la dataclass en su datastore asociado. Contiene los mismos atributos que la clase de datos, así como los valores de los datos y las propiedades y funciones específicas.

## Resumen

<code>.attributeName : any</code>	almacena el valor del atributo para la entidad
<code>.clone() : 4D.Entity</code>	crea en memoria una nueva entidad que hace referencia al mismo registro que la entidad original
<code>.diff( entityToCompare : 4D.Entity { ; attributesToCompare : Collection } ) : Collection</code>	compara el contenido de dos entidades y devuelve sus diferencias
<code>.drop( {mode : Integer} ) : Object</code>	elimina los datos contenidos en la entidad del datastore
<code>.first(): 4D.Entity</code>	devuelve una referencia a la entidad en primera posición de la selección de entidades a la que pertenece la entidad
<code>.fromObject( filler : Object )</code>	llena una entidad con el contenido <code>filler</code> .

Esta función modifica la entidad original.

El mapeo entre el objeto y la entidad se realiza sobre los nombres de los atributos:

- Si una propiedad del objeto no existe en la dataclass, se ignora.
- Los tipos de datos deben ser equivalentes. Si hay una diferencia de tipo entre el objeto y la dataclass, 4D intenta convertir los datos siempre que sea posible (ver [Convertir tipos de datos](#)), de lo contrario el atributo se deja sin tocar.
- La llave primaria puede darse tal cual o con una propiedad "\_\_KEY" (llenada con el valor de la llave primaria). Si no existe ya en la dataclass, la entidad se crea con el valor dado cuando se llama a `.save()`. Si no se da la llave primaria, se crea la entidad y se asigna el valor de la llave primaria con respecto a las reglas de la base de datos. El autoincremento sólo se calcula si la llave primaria es nula.

`filler` puede manejar una entidad relacionada bajo las siguientes condiciones:

- `filler` contiene la propia llave foránea, o
- `filler` contiene un objeto de propiedad con el mismo nombre que la entidad relacionada, que contiene una única propiedad denominada "\_\_KEY".
- si la entidad relacionada no existe, se ignora.

## Ejemplo

Con el siguiente objeto \$o:

```
{
    "firstName": "Mary",
    "lastName": "Smith",
    "salary": 36500,
    "birthDate": "1958-10-27T00:00:00.000Z",
    "woman": true,
    "managerID": 411,// relatedEntity dada con PK
    "employerID": 20 // relatedEntity dada con PK
}
```

El siguiente código creará una entidad con entidades relacionadas con el gerente y el empleador.

```
var $o : Object
var $entity : cs.EmpEntity
$entity:=ds.Emp.new()
$entity.fromObject($o)
$entity.save()
```

También puede utilizar una entidad relacionada dada como objeto:

```
{
    "firstName": "Marie",
    "lastName": "Lechat",
    "salary": 68400,
    "birthDate": "1971-09-03T00:00:00.000Z",
    "woman": false,
    "employer": {// relatedEntity dada como un objeto
        "__KEY": "21"
    },
    "manager": {// relatedEntity dada como un objeto
        "__KEY": "411"
    }
}
```

| .getDataClass() : 4D.DataClass

devuelve la clase de datos de la entidad | | .getKey( { mode : Integer } ) : Text  
.getKey( { mode : Integer } ) : Integer

devuelve el valor de la llave primaria | | Devuelve:

| | .getSelection() : 4D.EntitySelection

devuelve la selección de entidades a la que pertenece la entidad | | .getStamp() : Integer

devuelve el valor actual del sello de la entidad | | .indexOf( { entitySelection : 4D.EntitySelection } ) : Integer

devuelve la posición de la entidad en una selección de entidades | | .isNew() : Boolean

devuelve True si la entidad a la que se aplica acaba de ser creada y aún no ha sido guardada en el datastore | | .last() : 4D.Entity

| | .lock( { mode : Integer } ) : Object

pone un bloqueo pesimista en el registro referenciado por la entidad | | .next() : 4D.Entity

devuelve una referencia a la siguiente entidad en la selección de entidades a la que pertenece la entidad | | .previous() : 4D.Entity

devuelve una referencia a la entidad anterior en la selección de entidades a la que pertenece la entidad || [.reload\(\)](#)  
: Object

recarga el contenido de la entidad en memoria || [.save\( { mode : Integer } \)](#) : Object

guarda los cambios realizados en la entidad || [.toObject\(\)](#) : Object  
.toObject( *filterString* : Text { ; *options* : Integer } ) : Object  
.toObject( *filterCol* : Collection { ; *options* : Integer } ) : Object

devuelve un objeto que ha sido construido a partir de la entidad || [.touched\(\)](#) : Boolean

comprueba si un atributo de la entidad ha sido modificado o no desde que la entidad fue cargada en memoria o guardada || [.touchedAttributes\(\)](#) : Collection

devuelve los nombres de los atributos que han sido modificados desde que la entidad fue cargada en memoria || [.unlock\(\)](#) : Object

elimina el bloqueo pesimista del registro que coincide con la entidad |

## .attributeName

► Histórico

.attributeName : any

### Descripción

Todo atributo de la dataclass está disponible como una propiedad de una entidad, que almacena el valor del atributo para la entidad.

Los atributos dataclass también se pueden alcanzar utilizando la sintaxis alternativa con [ ].

El tipo de valor del atributo depende del tipo [kind](#) de atributo (relación o almacenamiento):

- Si el tipo de *attributeName* es storage: `.attributeName` devuelve un valor del mismo tipo que *attributeName*.
- Si el tipo de *attributeName* es relatedEntity: `.attributeName` devuelve la entidad relacionada. Los valores de la entidad relacionada están disponibles directamente a través de las propiedades en cascada, por ejemplo "myEntity.employer.employees[0].lastname".
- Si el tipo de *attributeName* es relatedEntities: `.attributeName` devuelve una nueva entity selection de entidades relacionadas. Se eliminan los duplicados (se devuelve una entity selection desordenada).

### Ejemplo

```
var $myEntity : cs.EmployeeEntity
$myEntity:=ds.Employee.new() //Crear una nueva entidad
$myEntity.name:="Dupont" // asignar 'Dupont' al atributo 'name'
$myEntity.firstname:="John" //asignar 'John' al atributo 'firstname'
$myEntity.save() //guardar la entidad
```

## .clone()

► Histórico

.clone() : 4D.Entity

Parámetros	Tipo		Descripción
Resultado	4D.Entity	<-	Nueva entidad que hace referencia al registro

### Descripción

La función `.clone()` crea en memoria una nueva entidad que hace referencia al mismo registro que la entidad original.

Esta función permite actualizar las entidades por separado.

Tenga en cuenta que toda modificación realizada a las entidades se guardará en el registro referenciado sólo cuando se ejecute la función `.save()`.

Esta función sólo puede utilizarse con entidades ya guardadas en la base de datos. No se puede llamar a una entidad recién creada (para la que `.isNew()` devuelve True).

## Ejemplo

```
var $emp; $empCloned : cs.EmployeeEntity  
$emp:=ds.Employee.get(672)  
$empCloned:=$emp.clone()  
  
$emp.lastName:="Smith" //Las actualizaciones realizadas en $emp no se realizan en $empCloned
```

## .diff()

► Histórico

`.diff( entityToCompare : 4D.Entity { ; attributesToCompare : Collection } ) : Collection`

Parámetros	Tipo		Descripción
entityToCompare	4D.Entity	->	Entidad a comparar con la entidad original
attributesToCompare	Collection	->	Nombre de los atributos a comparar
Resultado	Collection	<-	Diferencias entre las entidades

### Descripción

La función `.diff()` compara el contenido de dos entidades y devuelve sus diferencias.

En `entityToCompare`, pase la entidad a comparar con la entidad original.

En `attributesToCompare`, puede designar atributos específicos para comparar. Si se suministra, la comparación se realiza sólo en los atributos especificados. Si no se suministra, se devuelven todas las diferencias entre las entidades.

Las diferencias se devuelven como una colección de objetos cuyas propiedades son:

Nombre de propiedad	Tipo	Descripción
attributeName	Cadena	Nombre del atributo
value	cualquiera - Depende del tipo de atributo	Valor del atributo en la entidad
otherValue	cualquiera - Depende del tipo de atributo	Valor del atributo en <code>entityToCompare</code>

Sólo se incluyen en la colección los atributos con valores diferentes. Si no se encuentran diferencias, `.diff()` devuelve una colección vacía.

La función se aplica a las propiedades cuyo `kind` es `storage` o `relatedEntity`. En caso de que se haya actualizado una entidad relacionada (es decir, la llave foránea), el nombre de la entidad relacionada y su nombre de llave primaria se devuelven como propiedades `attributeName` (`value` y `otherValue` están vacíos para el nombre de la entidad relacionada).

Si una de las entidades comparadas es `Null`, se produce un error.

## Ejemplo 1

```

var $diff1; $diff2 : Collection
employee:=ds.Employee.query("ID=1001").first()
$clone:=employee.clone()
employee.firstName=="MARIE"
employee.lastName=="SOPHIE"
employee.salary==500
$diff1:=$clone.diff(employee) // Se devuelven todas las diferencias
$diff2:=$clone.diff(employee;New collection"firstName";"lastName"))
// Sólo se devuelven las diferencias en el firstName y lastName

```

\$diff1:

```

[
  {
    "attributeName": "firstName",
    "value": "Natasha",
    "otherValue": "MARIE"
  },
  {
    "attributeName": "lastName",
    "value": "Locke",
    "otherValue": "SOPHIE"
  },
  {
    "attributeName": "salary",
    "value": 66600,
    "otherValue": 500
  }
]
$diff2:

```

```

[
  {
    "attributeName": "firstName",
    "value": "Natasha",
    "otherValue": "MARIE"
  },
  {
    "attributeName": "lastName",
    "value": "Locke",
    "otherValue": "SOPHIE"
  }
]
```

Ejemplo 2

```

var vCompareResult1; vCompareResult2; vCompareResult3; $attributesToInspect : Collection
vCompareResult1:=New collection
vCompareResult2:=New collection
vCompareResult3:=New collection
$attributesToInspect:=New collection

$e1:=ds.Employee.get(636)
$e2:=ds.Employee.get(636)

$e1.firstName:=$e1.firstName+" update"
$e1.lastName:=$e1.lastName+" update"

$c:=ds.Company.get(117)
$e1.employer:=$c
$e2.salary:=100

$attributesToInspect.push("firstName")
$attributesToInspect.push("lastName")

vCompareResult1:=$e1.diff($e2)
vCompareResult2:=$e1.diff($e2;$attributesToInspect)
vCompareResult3:=$e1.diff($e2;$e1.touchedAttributes())

```

vCompareResult1 (se devuelven todas las diferencias):

```
[
{
  "attributeName": "firstName",
  "value": "Karla update",
  "otherValue": "Karla"
},
{
  "attributeName": "lastName",
  "value": "Marrero update",
  "otherValue": "Marrero"
},
{
  "attributeName": "salary",
  "value": 33500,
  "otherValue": 100
},
{
  "attributeName": "employerID",
  "value": 117,
  "otherValue": 118
},
{
  "attributeName": "employer",
  "value": "[object Entity]",// Entity 117 from Company
  "otherValue": "[object Entity]"// Entity 118 from Company
}
]
```

vCompareResult2 (sólo se devuelven las diferencias en \$attributesToInspect)

```
[
  {
    "attributeName": "firstName",
    "value": "Karla update",
    "otherValue": "Karla"
  },
  {
    "attributeName": "lastName",
    "value": "Marrero update",
    "otherValue": "Marrero"
  }
]
```

vCompareResult3 (sólo se devuelven las diferencias en atributos tocados \$e1)

```
[
  {
    "attributeName": "firstName",
    "value": "Karla update",
    "otherValue": "Karla"
  },
  {
    "attributeName": "lastName",
    "value": "Marrero update",
    "otherValue": "Marrero"
  },
  {
    "attributeName": "employerID",
    "value": 117,
    "otherValue": 118
  },
  {
    "attributeName": "employer",
    "value": "[object Entity]",// Entity 117 from Company
    "otherValue": "[object Entity]"// Entity 118 from Company
  }
]
```

## .drop()

► Histórico

.drop( {mode : Integer} ) : Object

Parámetros	Tipo		Descripción
mode	Integer	->	dk force drop if stamp changed : activa el soltar incluso si el sello ha cambiado
Resultado	Object	<-	Resultado de la operación soltar

### Descripción

La función `.drop()` elimina los datos contenidos en la entidad del datastore, de la tabla relacionada con su Dataclass. Tenga en cuenta que la entidad permanece en la memoria.

En una aplicación multiusuario o multiproceso, la función `.drop()` se ejecuta bajo un mecanismo de "[bloqueo optimista](#)", en el que un sello de bloqueo interno se incrementa automáticamente cada vez que se guarda el registro.

Por defecto, si se omite el parámetro `mode`, la función devolverá un error (ver más abajo) si la misma entidad fue modificada (es decir, el sello ha cambiado) por otro proceso o usuario en el ínterin.

De lo contrario, puede pasar la opción `dk force drop if stamp changed` en el parámetro *mode*: en este caso, la entidad se abandona aunque el sello haya cambiado (y la llave primaria siga siendo la misma).

## Resultado

El objeto devuelto por `.drop()` contiene las siguientes propiedades:

Propiedad		Tipo	Descripción
success		boolean	true si la acción de soltar tiene éxito, false en caso contrario.
			<i>Disponible sólo en caso de error:</i>
status(*)		number	Código de error, ver abajo
statusText(*)		text	Descripción del error, ver abajo
			<i>Disponible sólo en caso de error de bloqueo pesimista:</i>
LockKindText		text	"Locked by record"
lockInfo		object	Información sobre el origen del bloqueo
	task_id	number	Id del proceso
	user_name	text	Nombre de usuario de la sesión en la máquina
	user4d_alias	text	Alias de usuario si está definido por <code>SET USER ALIAS</code> , si no, nombre de usuario en el directorio 4D
	host_name	text	Nombre de la máquina
	task_name	text	Nombre del proceso
	client_version	text	
			<i>Disponible sólo en caso de error grave (un error grave puede ser intentar duplicar una llave primaria, disco lleno...):</i>
errors		collection of objects	
	message	text	Mensaje de error
	component signature	text	firma del componente interno (por ejemplo, "dmbg" significa el componente de la base)
	errCode	number	Código de error

(\*) Los siguientes valores pueden ser devueltos en las propiedades *status* y *statusText* del objeto *Result* en caso de error:

Constante	Valor	Comentario
<code>dk status entity does not exist anymore</code>	5	La entidad ya no existe en los datos. Este error puede ocurrir en los siguientes casos: <ul style="list-style-type: none"><li>• la entidad ha sido abandonada (el sello ha cambiado y el espacio de memoria está ahora libre)</li><li>• la entidad ha sido eliminada y sustituida por otra con otra llave primaria (el sello ha cambiado y una nueva entidad utiliza ahora el espacio de memoria). Cuando se utiliza <code>entity.drop()</code>, este error puede ser devuelto cuando se utiliza la opción <code>dk force drop if stamp changed</code>. Cuando se utiliza <code>entity.lock()</code>, este error puede ser devuelto cuando se utiliza la opción <code>dk reload if stamp changed</code></li></ul> statusText asociado: "La entidad ya no existe"
<code>dk status locked</code>	3	La entidad está bloqueada por un bloqueo pesimista. statusText asociado: "Ya está bloqueado"
<code>dk status serious error</code>	4	Un error grave es un error de base de datos de bajo nivel (por ejemplo, una llave duplicada), un error de hardware, etc. statusText asociado: "Otro error"
<code>dk status stamp has changed</code>	2	El valor del sello interno de la entidad no coincide con el de la entidad almacenada en los datos (bloqueo optimista). <ul style="list-style-type: none"><li>• con <code>.save()</code>: error sólo si no se utiliza la opción <code>dk auto merge</code></li><li>• con <code>.drop()</code>: error sólo si no se utiliza la opción <code>dk force drop if stamp changed</code></li><li>• con <code>.lock()</code>: error sólo si no se utiliza la opción <code>dk reload if stamp changed</code></li><li>• **statusText asociado**: "El sello ha cambiado"</li></ul>

## Ejemplo 1

Ejemplo sin la opción `dk force drop if stamp changed`:

```

var $employees : cs.EmployeeSelection
var $employee : cs.EmployeeEntity
var $status : Object
$employees:=ds.Employee.query("lastName=:1";"Smith")
$employee:=$employees.first()
$status:=$employee.drop()
Case of
    :($status.success)
        ALERT("You have dropped "+$employee.firstName+" "+$employee.lastName) //La entidad soltada permanece en la base de datos
    :($status.status=dk status stamp has changed)
        ALERT($status.statusText)
End case

```

## Ejemplo 2

Ejemplo con la opción `dk force drop if stamp changed`:

```

var $employees : cs.EmployeeSelection
var $employee : cs.EmployeeEntity
var $status : Object
$employees:=ds.Employee.query("lastName=:1";"Smith")
$employee:=$employees.first()
$status:=$employee.drop(dk force drop if stamp changed)
Case of
    :($status.success)
        ALERT("You have dropped "+$employee.firstName+" "+$employee.lastName) //La entidad soltada permanece
    :($status.status=dk status entity does not exist anymore)
        ALERT($status.statusText)
End case

```

## .first()

► Histórico

.first(): 4D.Entity

Parámetros	Tipo		Descripción
Resultado	4D.Entity	<-	Referencia a la primera entidad de una selección de entidades (Null si no se encuentra)

### Descripción

La función `.first()` devuelve una referencia a la entidad en primera posición de la selección de entidades a la que pertenece la entidad.

Si la entidad no pertenece a ninguna selección de entidades existente (es decir, `.getSelection( )` devuelve Null), la función devuelve un valor Null.

### Ejemplo

```

var $employees : cs.EmployeeSelection
var $employee; $firstEmployee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName = :1";"H@") //Esta selección de entidades contiene 3 entidades
$employee:=$employees[2]
$firstEmployee:=$employee.first() // $firstEmployee es la primera entidad de la selección de entidades $

```

## .fromObject()

► Histórico

.fromObject( *filler* : Object )

Parámetros	Tipo		Descripción
<i>filler</i>	Object	->	Objeto a partir del cual se llena la entidad

### Descripción

La función `.fromObject()` llena una entidad con el contenido *filler*.

Esta función modifica la entidad original.

El mapeo entre el objeto y la entidad se realiza sobre los nombres de los atributos:

- Si una propiedad del objeto no existe en la dataclass, se ignora.
- Los tipos de datos deben ser equivalentes. Si hay una diferencia de tipo entre el objeto y la dataclass, 4D intenta convertir los datos siempre que sea posible (ver [Convertir tipos de datos](#)), de lo contrario el atributo se deja sin tocar.
- La llave primaria puede darse tal cual o con una propiedad "\_\_KEY" (llenada con el valor de la llave primaria). Si no existe ya en la dataclass, la entidad se crea con el valor dado cuando se llama a `.save()`. Si no se da la llave primaria, se crea la entidad y se asigna el valor de la llave primaria con respecto a las reglas de la base de datos. El autoincremento sólo se calcula si la llave primaria es nula.

*filler* puede manejar una entidad relacionada bajo las siguientes condiciones:

- *filler* contiene la propia llave foránea, o
- *filler* contiene un objeto de propiedad con el mismo nombre que la entidad relacionada, que contiene una única propiedad denominada "\_\_KEY".
- si la entidad relacionada no existe, se ignora.

## Ejemplo

Con el siguiente objeto \$o:

```
{
  "firstName": "Mary",
  "lastName": "Smith",
  "salary": 36500,
  "birthDate": "1958-10-27T00:00:00.000Z",
  "woman": true,
  "managerID": 411,// relatedEntity dada con PK
  "employerID": 20 // relatedEntity dada con PK
}
```

El siguiente código creará una entidad con entidades relacionadas con el gerente y el empleador.

```
var $o : Object
var $entity : cs.EmpEntity
$entity:=ds.Emp.new()
$entity.fromObject($o)
$entity.save()
```

También puede utilizar una entidad relacionada dada como objeto:

```
{
  "firstName": "Marie",
  "lastName": "Lechat",
  "salary": 68400,
  "birthDate": "1971-09-03T00:00:00.000Z",
  "woman": false,
  "employer": {// relatedEntity dada como un objeto
    "__KEY": "21"
  },
  "manager": {// relatedEntity dada como un objeto
    "__KEY": "411"
  }
}
```

## .getDataClass()

.getDataClass() : 4D.DataClass

Parámetros	Tipo		Descripción
Resultado	4D.DataClass	<-	Objeto DataClass al que pertenece la entidad

## Descripción

La función `.getDataClass()` devuelve la clase de datos de la entidad. Esta función es útil al escribir código genérico.

## Ejemplo

El siguiente código genérico duplica cualquier entidad:

```
//método duplicate_entity
//duplicate_entity($entity)

#DECLARE($entity : 4D.Entity)
var $entityNew : 4D.Entity
var $status : Object

$entityNew:=$entity.getDataClass().new() //crea una nueva entidad en la dataclass padre
$entityNew.fromObject($entity.toObject()) //obtiene todos los atributos
$entityNew[$entity.getDataClass().getInfo().primaryKey]:=Null //restablece la llave primaria
$status:=$entityNew.save() //guarda la entidad duplicada
```

## .getKey()

► Histórico

.getKey( { mode : Integer } ) : Text  
.getKey( { mode : Integer } ) : Integer

Parámetros	Tipo		Descripción
mode	Integer	->	<code>dk key as string</code> : la llave primaria se devuelve como una cadena, sin importar el tipo de llave primaria
Resultado	Text	<-	Valor de la llave primaria de texto de la entidad
Resultado	Integer	<-	Valor de la llave primaria numérica de la entidad

## Descripción

La función `.getKey()` devuelve el valor de la llave primaria.

Las llaves primarias pueden ser números (enteros) o cadenas. Puede "forzar" que el valor de la llave primaria devuelto sea una cadena, sin importar el tipo de llave primaria real, pasando la opción `dk key as string` en el parámetro *mode*.

## Ejemplo

```
var $employees : cs.EmployeeSelection
var $employee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName=:1";"Smith")
$employee:=$employees[0]
ALERT("The primary key is "+$employee.getKey(dk key as string))
```

## .getRemoteContextAttributes()

► Histórico

Devuelve:

Parámetros	Tipo		Descripción
result	Texto	<-	Atributos de contexto vinculados a la entidad, separados por una coma

Modo avanzado: esta función está pensada para los desarrolladores que necesitan personalizar las funcionalidades por defecto de ORDA para configuraciones específicas. En la mayoría de los casos, no será necesario utilizarla.

Descripción

Devuelve:

Extracción de todas las propiedades de una `Entidad relacionada`:

Ejemplo

```
```4d
var $employee : cs.EmployeeEntity
var $status : Object
```

Ver también

Devuelve:

## .getSelection()

► Histórico

.getSelection(): 4D.EntitySelection

Parámetros	Tipo		Descripción
Resultado	4D.EntitySelection	<-	Entity selection a la que pertenece la entidad (nula si no se encuentra)

Descripción

La función `.getSelection()` devuelve la selección de entidades a la que pertenece la entidad.

Si la entidad no pertenece a una selección de entidades, la función devuelve Null.

Ejemplo

```
var $emp : cs.EmployeeEntity
var $employees; $employees2 : cs.EmployeeSelection
$emp:=ds.Employee.get(672) // Esta entidad no pertenece a ninguna selección de entidades
$employees:=$emp.getSelection() // $employees es Null

$employees2:=ds.Employee.query("lastName=:1";"Smith") //Esta selección de entidades contiene 6 entidades
$emp:=$employees2[0] // Esta entidad pertenece a una selección de entidades

ALERT("La entity selection contiene "+String($emp.getSelection().length)+" entidades")
```

## .getStamp()

► Histórico

.getStamp() : Integer

Parámetros	Tipo		Descripción
Resultado	Integer	<-	Sello de la entidad (0 si la entidad acaba de ser creada)

## Descripción

La función `.getStamp()` devuelve el valor actual del sello de la entidad.

El sello interno se incrementa automáticamente en 4D cada vez que se guarda la entidad. Gestiona los accesos y modificaciones concurrentes de los usuarios a las mismas entidades (ver [Bloqueo de entidades](#)).

Para una entidad nueva (nunca guardada), la función devuelve 0. Para saber si una entidad acaba de ser creada, se recomienda utilizar `.isNew()`.

## Ejemplo

```
var $entity : cs.EmployeeEntity
var $stamp : Integer

$entity:=ds.Employee.new()
$entity.lastname:="Smith"
$entity.save()
$stamp:=$entity.getStamp() // $stamp=1

$entity.lastname:="Wesson"
$entity.save()
$stamp:=$entity.getStamp() // $stamp=2
```

## .indexOf()

► Histórico

.indexOf( { entitySelection : 4D.EntitySelection } ) : Integer

Parámetros	Tipo		Descripción
entitySelection	4D.EntitySelection	->	La posición de la entidad se da en función de esta selección de entidades
Resultado	Integer	<-	Posición de la entidad en una selección de entidades

## Descripción

La función `.indexOf()` devuelve la posición de la entidad en una selección de entidades.

Por defecto, si se omite el parámetro `entitySelection`, la función devuelve la posición de la entidad dentro de su propia selección de entidades. En caso contrario, devuelve la posición de la entidad dentro de la `entitySelection` especificada.

El valor resultante se incluye entre 0 y la longitud de la selección de entidades -1.

- Si la entidad no tiene una selección de entidad o no pertenece a `entidadSelección`, la función devuelve -1.
- Si `entitySelection` es Null o no pertenece a la misma clase de datos que la entidad, se produce un error.

## Ejemplo

```

var $employees : cs.EmployeeSelection
var $employee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName = :1";"H@") //Esta entity selection contiene 3 entidades
$employee:=$employees[1] //Esta entidad pertenece a una entity selection
ALERT("El índice de la entidad en su propia selección de entidades es "+String($employee.indexOf())) //

$employee:=ds.Employee.get(725) //Esta entidad no pertenece a una selección de entidades
ALERT("El índice de la entidad es "+String($employee.indexOf())) // -1

```

## .isNew()

► Histórico

.isNew() : Boolean

Parámetros	Tipo		Descripción
Resultado	Booleano	<-	True si la entidad acaba de ser creada y aún no se ha guardado. En caso contrario, False.

### Descripción

La función `.isNew()` devuelve True si la entidad a la que se aplica acaba de ser creada y aún no ha sido guardada en el datastore. En caso contrario, devuelve False.

### Ejemplo

```

var $emp : cs.EmployeeEntity

$emp:=ds.Employee.new()

If($emp.isNew())
    ALERT("This is a new entity")
End if

```

## .last()

► Histórico

.last() : 4D.Entity

Parámetros	Tipo		Descripción
Resultado	4D.Entity	<-	Referencia a la última entidad de una selección de entidades (Null si no se encuentra)

### Descripción

La función `.last()` devuelve una referencia a la entidad en la última posición de la selección de entidades a la que pertenece la entidad.

Si la entidad no pertenece a ninguna selección de entidades existente (es decir, `.getSelection()` devuelve Null), la función devuelve un valor Null.

### Ejemplo

```

var $employees : cs.EmployeeSelection
var $employee; $lastEmployee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName = :1";"H@") //Esta selección de entidades contiene 3 entidades
$employee:=$employees[0]
$lastEmployee:=$employee.last() // $lastEmployee es la última entidad de la selección de entidades $employees

```

## .lock()

► Histórico

`.lock( { mode : Integer } ) : Object`

Parámetros	Tipo		Descripción
mode	Integer	->	<code>dk reload if stamp changed</code> : recargar antes de bloquear si el sello ha cambiado
Resultado	Objeto	<-	Resultado de la operación de bloqueo

### Descripción

La función `.lock()` pone un bloqueo pesimista en el registro referenciado por la entidad. El [bloqueo se establece](#) para un registro y todas las referencias de la entidad en el proceso actual.

Otros procesos verán este registro como bloqueado (la propiedad `result.success` contendrá `False` si intentan bloquear la misma entidad usando esta función). Sólo las funciones ejecutadas en la sesión de "bloqueo" pueden editar y guardar los atributos de la entidad. La entidad puede ser cargada como de sólo lectura por otras sesiones, pero no podrán introducir y guardar valores.

El objeto devuelto por `<.unlock()>` contiene la siguiente propiedad:

- cuando la función `unlock()` se llama en una entidad correspondiente en el mismo proceso
- automáticamente, cuando ya no es referenciado por ninguna entidad en la memoria. Por ejemplo, si el bloqueo se pone sólo en una referencia local de una entidad, la entidad se desbloquea cuando la función termina. Mientras haya referencias a la entidad en la memoria, el registro permanece bloqueado.

Una entidad también puede ser [bloqueada por una sesión REST](#), en cuyo caso sólo puede ser desbloqueada por la sesión.

Por defecto, si se omite el parámetro `mode`, la función devolverá un error (ver más abajo) si la misma entidad fue modificada (es decir, el sello ha cambiado) por otro proceso o usuario en el interin.

De lo contrario, puede pasar la opción `dk reload if stamp changed` en el parámetro `mode`: en este caso, no se devuelve error y la entidad se recarga cuando el sello cambia (si la entidad aún existe y la llave primaria sigue siendo la misma).

### Resultado

El objeto devuelto por `.lock( )` contiene las siguientes propiedades:

Propiedad		Tipo	Descripción
success		boolean	true si la acción de bloqueo tiene éxito (o si la entidad ya está bloqueada en el proceso actual), false en caso contrario.  <i>Disponible sólo si se utiliza la opción <code>dk reload if stamp changed</code>:</i>
wasReloaded		boolean	true si la entidad fue recargada con éxito, false en caso contrario.  <i>Disponible sólo en caso de error:</i>
status(*)		number	Código de error, ver abajo
statusText(*)		text	Descripción del error, ver abajo  <i>Disponible sólo en caso de error de bloqueo pesimista:</i>
lockKindText		text	"Locked by record" si está bloqueado por un proceso 4D, "Locked by session" si está bloqueado por una sesión REST
lockInfo		object	Información sobre el origen del bloqueo. Las propiedades devueltas dependen del origen del bloqueo (proceso 4D o sesión REST).  <i>Disponible sólo para un bloqueo por proceso 4D:</i>
	task_id	number	ID del Proceso
	user_name	text	Nombre de usuario de la sesión en la máquina
	user4d_alias	texto	Nombre o alias del usuario 4D
	user4d_id	number	ID del usuario en el directorio de la base de datos 4D
	host_name	text	Nombre de la máquina
	task_name	text	Nombre del proceso
	client_version	texto	Versión del cliente
			<i>Disponible sólo para un bloqueo por sesión REST:</i>
	host	text	URL que bloqueó la entidad (por ejemplo, " <a href="http://www.myserver.com">www.myserver.com</a> ")
	IPAddr	text	Dirección IP del bloqueo (por ejemplo: "127.0.0.1")
	userAgent	texto	userAgent del origin del bloqueo (ej: "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36")
			<i>Disponible sólo en caso de error grave (la llave primaria ya existe, el disco está lleno...):</i>
errors		collection of objects	
	message	texto	Mensaje de error
	component signature	texto	firma del componente interno (por ejemplo, "dmbg" significa el componente de la base)
	errCode	number	Código de error

(\*) Los siguientes valores pueden ser devueltos en las propiedades *status* y *statusText* del objeto *Result* en caso de error:

Constante	Valor	Comentario
<code>dk status entity does not exist anymore</code>	5	<p>La entidad ya no existe en los datos. Este error puede ocurrir en los siguientes casos:</p> <ul style="list-style-type: none"> <li>• la entidad ha sido abandonada (el sello ha cambiado y el espacio de memoria está ahora libre)</li> <li>• la entidad ha sido eliminada y sustituida por otra con otra llave primaria (el sello ha cambiado y una nueva entidad utiliza ahora el espacio de memoria). Cuando se utiliza <code>.drop()</code>, este error puede devolverse cuando se utiliza la opción <code>dk force drop if stamp changed</code>. Cuando se utiliza <code>.lock()</code>, este error puede ser devuelto cuando se utiliza la opción <code>dk reload if stamp changed</code></li> </ul> <p>statusText asociado: "La entidad ya no existe"</p>
<code>dk status locked</code>	3	<p>La entidad está bloqueada por un bloqueo pesimista.</p> <p>statusText asociado: "Ya bloqueado"</p>
<code>dk status serious error</code>	4	<p>Un error grave es un error de base de datos de bajo nivel (por ejemplo, una llave duplicada), un error de hardware, etc.</p> <p>statusText asociado: "Otro error"</p>
<code>dk status stamp has changed</code>	2	<p>El valor del sello interno de la entidad no coincide con el de la entidad almacenada en los datos (bloqueo optimista).</p> <ul style="list-style-type: none"> <li>• con <code>.save()</code>: error sólo si no se utiliza la opción <code>dk auto merge</code></li> <li>• con <code>.drop()</code>: error sólo si no se utiliza la opción <code>dk force drop if stamp changed</code></li> <li>• con <code>.lock()</code>: error sólo si no se utiliza la opción <code>dk reload if stamp changed</code></li> </ul> <p>statusText asociado: "El sello ha cambiado"</p>

## Ejemplo 1

Ejemplo con error:

```

var $employee : cs.EmployeeEntity
var $status : Object
$employee:=ds.Employee.get(716)
$status:=$employee.lock()
Case of
  :($status.success)
    ALERT("You have locked "+$employee.firstName+" "+$employee.lastName)
  :($status.status=dk status stamp has changed)
    ALERT($status.statusText)
End case
  
```

## Ejemplo 2

Ejemplo con la opción `dk reload if stamp changed`:

```

var $employee : cs.EmployeeEntity
var $status : Object
$employee:=ds.Employee.get(717)
$status:=$employee.lock(dk reload if stamp changed)
Case of
    :($status.success)
        ALERT("You have locked "+$employee.firstName+" "+$employee.lastName)
    :($status.status=dk status entity does not exist anymore)
        ALERT($status.statusText)
End case

```

## .next()

► Histórico

.next() : 4D.Entity

Parámetros	Tipo		Descripción
Resultado	4D.Entity	<-	Referencia a la siguiente entidad en la selección de entidades (Null si no se encuentra)

### Descripción

La función `.next()` devuelve una referencia a la siguiente entidad en la selección de entidades a la que pertenece la entidad.

Si la entidad no pertenece a ninguna selección de entidades existente (es decir, `.getSelection()` devuelve Null), la función devuelve un valor Null.

Si no hay una entidad siguiente válida en la selección de entidades (es decir, se encuentra en la última entidad de la selección), la función devuelve Null. Si la siguiente entidad ha sido descartada, la función devuelve la siguiente entidad válida (y eventualmente Null).

### Ejemplo

```

var $employees : cs.EmployeeSelection
var $employee; $nextEmployee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName = :1;"H@") //Esta selección de entidades contiene 3 entidades
$employee:=$employees[0]
$nextEmployee:=$employee.next() // $nextEmployee es la segunda entidad de la selección de entidades $emp

```

## .previous()

► Histórico

.previous() : 4D.Entity

Parámetros	Tipo		Descripción
Resultado	4D.Entity	<-	Referencia a la entidad anterior en la selección de entidades (Null si no se encuentra)

### Descripción

La función `.previous()` devuelve una referencia a la entidad anterior en la selección de entidades a la que pertenece la entidad.

Si la entidad no pertenece a ninguna selección de entidades existente (es decir, `.getSelection()` devuelve Null), la

función devuelve un valor Null.

Si no hay una entidad anterior válida en la selección de entidades (es decir, se encuentra en la primera entidad de la selección), la función devuelve Null. Si la entidad anterior ha sido soltada, la función devuelve la entidad válida anterior (y eventualmente Null).

## Ejemplo

```
var $employees : cs.EmployeeSelection
var $employee; $previousEmployee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName = :1";"H@") //Esta selección de entidades contiene 3 entidades
$employee:=$employees[1]
$previousEmployee:=$employee.previous() // $previousEmployee es la primera entidad de la selección de en
```

## .reload( )

► Histórico

.reload() : Object

Parámetros	Tipo		Descripción
Resultado	Objeto	<-	Objeto de estado

### Descripción

La función `.reload()` recarga el contenido de la entidad en memoria, según la información almacenada en la tabla relacionada con la dataclass en el datastore. La recarga se realiza sólo si la entidad sigue existiendo con la misma llave primaria.

### Resultado

El objeto devuelto por `.reload( )` contiene las siguientes propiedades:

Propiedad	Tipo	Descripción
success	boolean	True si la acción de recargar tiene éxito, False en caso contrario. <i>Disponible sólo en caso de error:</i>
status(*)	number	Código de error, ver abajo
statusText(*)	text	Descripción del error, ver abajo

(\*) Los siguientes valores pueden ser devueltos en las propiedades `status` y `statusText` del objeto `Result` en caso de error:

Constante	Valor	Comentario
dk status entity does not exist anymore	5	<p>La entidad ya no existe en los datos. Este error puede ocurrir en los siguientes casos:</p> <ul style="list-style-type: none"> <li>• la entidad ha sido abandonada (el sello ha cambiado y el espacio de memoria está ahora libre)</li> <li>• la entidad ha sido eliminada y sustituida por otra con otra llave primaria (el sello ha cambiado y una nueva entidad utiliza ahora el espacio de memoria). la entidad ha sido eliminada y sustituida por otra con otra llave primaria (el sello ha cambiado y una nueva entidad utiliza ahora el espacio de memoria). Cuando se utiliza <code>.lock()</code>, este error puede ser devuelto cuando se utiliza la opción <code>dk reload if stamp changed</code></li> </ul> <p><i>statusText asociado: "La entidad ya no existe"</i></p>
dk status serious error	4	<p>Un error grave es un error de base de datos de bajo nivel (por ejemplo, una llave duplicada), un error de hardware, etc.</p> <p><i>statusText asociado: "Otro error"</i></p>

## Ejemplo

```

var $employee : cs.EmployeeEntity
var $employees : cs.EmployeeSelection
var $result : Object

$employees:=ds.Employee.query("lastName=:1";"Hollis")
$employee:=$employees[0]
$employee.firstName:="Mary"
$result:=$employee.reload()
Case of
    :($result.success)
        ALERT("Reload has been done")
    :($result.status=dk status entity does not exist anymore)
        ALERT("The entity has been dropped")
End case

```

## .save()

► Histórico

`.save( { mode : Integer } ) : Object`

Parámetros	Tipo		Descripción
mode	Integer	->	<code>dk auto merge</code> : activa el modo de fusión automática
Resultado	Objeto	<-	Resultado de la operación guardar

## Descripción

La función `.save()` guarda los cambios realizados en la entidad en la tabla relacionada con su dataClass. Debe llamar a este método después de crear o modificar una entidad si quiere guardar los cambios realizados en ella.

La operación de guardar se ejecuta sólo si se ha "tocado" al menos un atributo de la entidad (ver las funciones `.touched()` y `.touchedAttributes()`). En caso contrario, la función no hace nada (no se llama al activador).

En una aplicación multiusuario o multiproceso, la función `.save()` se ejecuta bajo un mecanismo de "bloqueo optimista", en el que un sello de bloqueo interno se incrementa automáticamente cada vez que se guarda el registro.

Por defecto, si se omite el parámetro `mode`, el método devolverá un error (ver más abajo) siempre que la misma entidad haya sido modificada por otro proceso o usuario mientras tanto, sin importar el atributo o atributos modificados.

En caso contrario, se puede pasar la opción `dk auto merge` en el parámetro `mode`: cuando el modo de fusión automática está activado, una modificación realizada simultáneamente por otro proceso/usuario en la misma entidad pero en un atributo diferente no dará lugar a un error. Los datos resultantes guardados en la entidad serán la combinación (la "fusión") de todas las modificaciones no concurrentes (si se aplicaron modificaciones al mismo atributo, el guardado falla y se devuelve un error, incluso con el modo de fusión automática).

El modo de fusión automática no está disponible para los atributos de tipo Imagen, Objeto y Texto cuando se almacenan fuera del registro. Los cambios concurrentes en estos atributos darán lugar a un error `dk status stamp has changed`.

## Resultado

El objeto devuelto por `.save()` contiene las siguientes propiedades:

Propiedad		Tipo	Descripción
success		boolean	True si la acción guardar tiene éxito, false en caso contrario.
			<i>Disponible sólo si se utiliza la opción dk auto merge :</i>
autoMerged		boolean	True si se ha realizado una fusión automática, False en caso contrario.
			<i>Disponible sólo en caso de error:</i>
status		number	Código de error, <a href="#">ver abajo</a>
statusText		text	Descripción del error, <a href="#">ver más abajo</a>
			<i>Disponible sólo en caso de bloqueo pesimista :</i>
lockKindText		text	"Locked by record"
lockInfo		object	Información sobre el origen del bloqueo
	task_id	number	Id del proceso
	user_name	text	Nombre de usuario de la sesión en la máquina
	user4d_alias	text	Alias de usuario si está definido por <code>SET USER ALIAS</code> , si no, nombre de usuario en el directorio 4D
	host_name	text	Nombre de la máquina
	task_name	text	Nombre del proceso
	client_version	text	
			<i>Disponible sólo en caso de error grave (error grave - puede ser intentar duplicar una llave primaria, disco lleno...):</i>
errors		collection of objects	
	message	text	Mensaje de error
	componentSignature	text	Firma del componente interno (por ejemplo, "dmbg" significa el componente de la base)
	errCode	number	Código de error

## status y statusText

Los siguientes valores pueden ser devueltos en las propiedades `status` y `statusText` del objeto Result en caso de error:

Constante	Valor	Comentario
<code>dk status automerge failed</code>	6	(Sólo si se utiliza la opción <code>dk auto merge</code> ) La opción de fusión automática falló al guardar la entidad. statusText asociado: "Fallo de la fusión automática"
<code>dk status entity does not exist anymore</code>	5	La entidad ya no existe en los datos. Este error puede ocurrir en los siguientes casos: <ul style="list-style-type: none"> <li>• la entidad ha sido abandonada (el sello ha cambiado y el espacio de memoria está ahora libre)</li> <li>• la entidad ha sido eliminada y sustituida por otra con otra llave primaria (el sello ha cambiado y una nueva entidad utiliza ahora el espacio de memoria). la entidad ha sido eliminada y sustituida por otra con otra llave primaria (el sello ha cambiado y una nueva entidad utiliza ahora el espacio de memoria). Cuando se utiliza <code>.lock()</code>, este error puede ser devuelto cuando se utiliza la opción <code>dk reload if stamp changed</code></li> </ul> statusText asociado: "La entidad ya no existe"
<code>dk status locked</code>	3	La entidad está bloqueada por un bloqueo pesimista. statusText asociado: "Ya bloqueado"
<code>dk status serious error</code>	4	Un error grave es un error de base de datos de bajo nivel (por ejemplo, una llave duplicada), un error de hardware, etc. statusText asociado: "Otro error"
<code>dk status stamp has changed</code>	2	El valor del sello interno de la entidad no coincide con el de la entidad almacenada en los datos (bloqueo optimista). <ul style="list-style-type: none"> <li>• con <code>.save()</code>: error sólo si no se utiliza la opción <code>dk auto merge</code></li> <li>• con <code>.drop()</code>: error sólo si no se utiliza la opción <code>dk force drop if stamp changed</code></li> <li>• con <code>.lock()</code>: error sólo si no se utiliza la opción <code>dk reload if stamp changed</code></li> </ul> statusText asociado: "El sello ha cambiado"

## Ejemplo 1

Crear una nueva entidad:

```
var $status : Object
var $employee : cs.EmployeeEntity
$employee:=ds.Employee.new()
$employee.firstName:="Mary"
$employee.lastName:="Smith"
$status:=$employee.save()
If($status.success)
  ALERT("Employee created")
End if
```

## Ejemplo 2

Actualización de una entidad sin la opción `dk auto merge`:

```

var $status : Object
var $employee : cs.EmployeeEntity
var $employees : cs.EmployeeSelection
$employees:=ds.Employee.query("lastName=:1";"Smith")
$employee:=$employees.first()
$employee.lastName:="Mac Arthur"
$status:=$employee.save()
Case of
    :($status.success)
        ALERT("Employee updated")
    :($status.status=dk status stamp has changed)
        ALERT($status.statusText)
End case

```

## Ejemplo 3

Actualización de una entidad con la opción `dk auto merge`:

```

var $status : Object

var $employee : cs.EmployeeEntity
var $employees : cs.EmployeeSelection

$employees:=ds.Employee.query("lastName=:1";"Smith")
$employee:=$employees.first()
$employee.lastName:="Mac Arthur"
$status:=$employee.save(dk auto merge)
Case of
    :($status.success)
        ALERT("Employee updated")
    :($status.status=dk status automerge failed)
        ALERT($status.statusText)
End case

```

## .toObject()

► Histórico

`.toObject() : Object`  
`.toObject( filterString : Text { ; options : Integer} ) : Object`  
`.toObject( filterCol : Collection { ; options : Integer } ) : Object`

Parámetros	Tipo		Descripción
filterString	Texto	->	Atributo(s) a extraer (cadena separada por comas)
filterCol	Collection	->	Colección de atributos a extraer
options	Integer	->	<code>dk with primary key</code> : añade la propiedad <code>_KEY</code> ; <code>dk with stamp</code> : añade la propiedad <code>_STAMP</code>
Resultado	Objeto	<-	Objeto creado a partir de la entidad

### Descripción

La función `.toObject()` devuelve un objeto que ha sido construido a partir de la entidad. Los nombres de las propiedades en el objeto coinciden con los nombres de los atributos de la entidad.

Si no se especifica ningún filtro, o si el parámetro `filterString` contiene una cadena vacía o `"*"`, el objeto devuelto contendrá:

- todos los atributos de la entidad de almacenamiento

- atributos de la `relatedEntity` `kind`: se obtiene una propiedad con el mismo nombre que la entidad relacionada (nombre del enlace muchos-a-uno). El atributo se extrae con la forma simple.
- atributos donde `kind` es `relatedEntities` : no se devuelve el atributo.

En el primer parámetro, se pasa el atributo o atributos de la entidad a extraer. Puede pasar:

- `filterString`: una cadena con rutas de propiedades separadas por comas: "propertyPath1, propertyPath2, ...", o
- `filterCol`: una colección de cadenas: ["propertyPath1", "propertyPath2";...]

Si se especifica un filtro para los atributos de la `relatedEntity` `kind`:

- `propertyPath = "relatedEntity"` -> se extrae de forma sencilla: un objeto con la propiedad `__KEY` (llave primaria).
- `propertyPath = "relatedEntity.*"` -> se extraen todas las propiedades
- `propertyPath = "relatedEntity.propertyName1; relatedEntity.propertyName2; ..."` -> sólo se extraen esas propiedades

Si se especifica un filtro para los atributos de las `relatedEntities` `kind`:

- `propertyPath = "relatedEntities.*"` -> se extraen todas las propiedades
- `propertyPath = "relatedEntities.propertyName1; relatedEntities.propertyName2; ..."` -> sólo se extraen esas propiedades

En el parámetro `options` se puede pasar el selector `ddk with primary key` y/o `dk with stamp` para añadir las llaves primarias de la entidad y/o los sellos en los objetos extraídos.

## Ejemplo 1

En todos los ejemplos de esta sección se utilizará la siguiente estructura:

Sin parámetro de filtro:

```
employeeObject:=employeeSelected.toObject()
```

Devuelve:

```
{
  "ID": 413,
  "firstName": "Greg",
  "lastName": "Wahl",
  "salary": 0,
  "birthDate": "1963-02-01T00:00:00.000Z",
  "woman": false,
  "managerID": 412,
  "employerID": 20,
  "photo": "[object Picture]",
  "extra": null,
  "employer": { // relatedEntity extracted with simple form
    "__KEY": 20
  },
  "manager": {
    "__KEY": 412
  }
}
```

## Ejemplo 2

Extraer la llave primaria y el sello:

```
employeeObject:=employeeSelected.toObject("");dk with primary key+dk with stamp)
```

Devuelve:

```
{  
    "__KEY": 413,  
    "__STAMP": 1,  
    "ID": 413,  
    "firstName": "Greg",  
    "lastName": "Wahl",  
    "salary": 0,  
    "birthDate": "1963-02-01T00:00:00.000Z",  
    "woman": false,  
    "managerID": 412,  
    "employerID": 20,  
    "photo": "[object Picture]",  
    "extra": null,  
    "employer": {  
        "__KEY": 20  
    },  
    "manager": {  
        "__KEY": 412  
    }  
}
```

### Ejemplo 3

Desplegando todas las propiedades de `relatedEntities` :

```
employeeObject:=employeeSelected.toObject("directReports.*")
```

```
{
  "directReports": [
    {
      "ID": 418,
      "firstName": "Lorena",
      "lastName": "Boothe",
      "salary": 44800,
      "birthDate": "1970-10-02T00:00:00.000Z",
      "woman": true,
      "managerID": 413,
      "employerID": 20,
      "photo": "[object Picture]",
      "extra": null,
      "employer": {
        "__KEY": 20
      },
      "manager": {
        "__KEY": 413
      }
    },
    {
      "ID": 419,
      "firstName": "Drew",
      "lastName": "Caudill",
      "salary": 41000,
      "birthDate": "2030-01-12T00:00:00.000Z",
      "woman": false,
      "managerID": 413,
      "employerID": 20,
      "photo": "[object Picture]",
      "extra": null,
      "employer": {
        "__KEY": 20
      },
      "manager": {
        "__KEY": 413
      }
    },
    {
      "ID": 420,
      "firstName": "Nathan",
      "lastName": "Gomes",
      "salary": 46300,
      "birthDate": "2010-05-29T00:00:00.000Z",
      "woman": false,
      "managerID": 413,
      "employerID": 20,
      "photo": "[object Picture]",
      "extra": null,
      "employer": {
        "__KEY": 20
      },
      "manager": {
        "__KEY": 413
      }
    }
  ]
}
}
```

#### Ejemplo 4

Extracción de algunas propiedades de `relatedEntities`:

```
employeeObject:=employeeSelected.toObject("firstName, directReports.lastName")
```

Devuelve:

```
{  
    "firstName": "Greg",  
    "directReports": [  
        {  
            "lastName": "Boothe"  
        },  
        {  
            "lastName": "Caudill"  
        },  
        {  
            "lastName": "Gomes"  
        }  
    ]  
}
```

## Ejemplo 5

Obtenga una `relatedEntity` en un formulario simple:

```
$coll:=New collection("firstName";"employer")  
employeeObject:=employeeSelected.toObject($coll)
```

Devuelve:

```
{  
    "firstName": "Greg",  
    "employer": {  
        "__KEY": 20  
    }  
}
```

## Ejemplo 6

Extracción de todas las propiedades de una `Entidad relacionada`:

```
employeeObject:=employeeSelected.toObject("employer.*")
```

Devuelve:

```
{  
    "employer": {  
        "ID": 20,  
        "name": "India Astral Secretary",  
        "creationDate": "1984-08-25T00:00:00.000Z",  
        "revenues": 12000000,  
        "extra": null  
    }  
}
```

## Ejemplo 7

Extracción de algunas propiedades de una Entidad relacionada :

```
$col:=New collection  
$col.push("employer.name")  
$col.push("employer.revenues")  
employeeObject:=employeeSelected.toObject($col)
```

Devuelve:

```
{  
    "employer": {  
        "name": "India Astral Secretary",  
        "revenues": 12000000  
    }  
}
```

## .touched( )

► Histórico

.touched() : Boolean

Parámetros	Tipo		Descripción
Resultado	Booleano	<-	True si se ha modificado al menos un atributo de la entidad y aún no se ha guardado, si no, False

### Descripción

La función `.touched()` comprueba si un atributo de la entidad ha sido modificado o no desde que la entidad fue cargada en memoria o guardada.

Si un atributo ha sido modificado o calculado, la función devuelve True, en caso contrario devuelve False. Puede utilizar esta función para determinar si necesita guardar la entidad.

Esta función devuelve False para una nueva entidad que acaba de ser creada (con `.new()`). Tenga en cuenta, sin embargo, que si utiliza una función que calcula un atributo de la entidad, la función `.touched()` devolverá entonces True. Por ejemplo, si se llama a `.getKey()` para calcular la llave primaria, `.touched()` devuelve True.

### Ejemplo

En este ejemplo, comprobamos si es necesario guardar la entidad:

```
var $emp : cs.EmployeeEntity  
$emp:=ds.Employee.get(672)  
$emp.firstName:=$emp.firstName //Aunque se actualice con el mismo valor, el atributo se marca como tocado  
If($emp.touched()) //si se ha modificado al menos uno de los atributos  
    $emp.save()  
End if // de lo contrario, no es necesario guardar la entidad
```

## .touchedAttributes( )

► Histórico

.touchedAttributes() : Collection

Parámetros	Tipo		Descripción
Resultado	Collection	<-	Nombres de atributos tocados, o colección vacía

## Descripción

La función `.touchedAttributes()` devuelve los nombres de los atributos que han sido modificados desde que la entidad fue cargada en memoria.

Esto se aplica a los atributos `kind storage` o `relatedEntity`.

En el caso de que se haya tocado una entidad relacionada (es decir, la llave externa), se devuelve el nombre de la entidad relacionada y el nombre de su llave primaria.

Si no se ha tocado ningún atributo de entidad, el método devuelve una colección vacía.

## Ejemplo 1

```
var $touchedAttributes : Collection
var $emp : cs.EmployeeEntity

$touchedAttributes:=New collection
$emp:=ds.Employee.get(725)
$emp.firstName:=$emp.firstName //Aunque se actualice con el mismo valor, el atributo se marca como toca
$emp.lastName:="Martin"
$touchedAttributes:=$emp.touchedAttributes()
//$touchedAttributes: ["firstName","lastName"]
```

## Ejemplo 2

```
var $touchedAttributes : Collection
var $emp : cs.EmployeeEntity
var $company : cs.CompanyEntity

$touchedAttributes:=New collection

$emp:=ds.Employee.get(672)
$emp.firstName:=$emp.firstName
$emp.lastName:="Martin"

$company:=ds.Company.get(121)
$emp.employer:=$company

$touchedAttributes:=$emp.touchedAttributes()

//collection $touchedAttributes: ["firstName","lastName","employer","employerID"]
```

En este caso:

- `firstName` y `lastName` tienen un tipo `storage`
- `employer` tiene un tipo `relatedEntity`
- `employerID` es la llave extranjera de la entidad relacionada con el empleador

## .unlock()

► Histórico

`.unlock() : Object`

Parámetros	Tipo		Descripción
Resultado	Objeto	<-	Objeto de estado

## Descripción

La función `.unlock()` elimina el bloqueo pesimista del registro que coincide con la entidad en el datastore y la tabla relacionada con su dataclass.

Para más información, consulte la sección [Entity locking](#).

Un registro se desbloquea automáticamente cuando ya no es referenciado por ninguna entidad en el proceso de bloqueo (por ejemplo: si el bloqueo se pone sólo en una referencia local de una entidad, la entidad y, por tanto, el registro se desbloquea cuando el proceso termina).

Cuando un registro se bloquea, debe desbloquearse desde el proceso de bloqueo y en la referencia de la entidad que puso el bloqueo. Por ejemplo:

```
$e1:=ds.Emp.all()[0]
$e2:=ds.Emp.all()[0]
$res:=$e1.lock() // $res.success=true
$res:=$e2.unlock() // $res.success=false
$res:=$e1.unlock() // $res.success=true
```

## Resultado

El objeto devuelto por `<.unlock()>` contiene la siguiente propiedad:

Propiedad	Tipo	Descripción
success	Boolean	True si la acción de desbloquear tiene éxito, False en caso contrario. Si el desbloqueo se realiza en una entidad abandonada, en un registro no bloqueado o en un registro bloqueado por otro proceso o entidad, el éxito es False.

## Ejemplo

```
var $employee : cs.EmployeeEntity
var $status : Object

$employee:=ds.Employee.get(725)
$status:=$employee.lock()
... //processing
$status:=$employee.unlock()
If($status.success)
    ALERT("The entity is now unlocked")
End if
```

# EntitySelection

Una entity selection es un objeto que contiene una o más referencias a [entidades](#) pertenecientes a la misma [Dataclass](#). Una entity selection puede contener 0, 1 o X entidades de la dataclass -- donde X puede representar el número total de entidades contenidas en la dataclass.

Las selecciones de entidades pueden crearse a partir de selecciones existentes utilizando varias funciones de la clase [DataClass](#) tales como [.all\(\)](#) o [.query\(\)](#), o de la clase [EntityClass](#) misma, tal como [.and\(\)](#) o [orderBy\(\)](#). También puede crear entity selections vacías utilizando la función [dataClass.newSelection\(\)](#) o el comando [Create new selection](#).

## Resumen

### [\[index\] : 4D.Entity](#)

permite acceder a las entidades dentro de la selección de entidades utilizando la sintaxis de colección estándar

### [.attributeName : Collection](#)

### [.attributeName : 4D.EntitySelection](#)

una "proyección" de los valores del atributo en la entity selection

### [.add\( entity : 4D.Entity \) : 4D.EntitySelection](#)

añade la *entity* especificada a la entity selection y devuelve la selección de entidades modificada

### [.and\( entity : 4D.Entity \) : 4D.EntitySelection](#)

### [.and\( entitySelection : 4D.EntitySelection \) : 4D.EntitySelection](#)

combina la entity selection con el parámetro *entity* o *entitySelection* utilizando el operador lógico AND

### [.average\( attributePath : Text \) : Real](#)

devuelve la media aritmética (promedio) de todos los valores no nulos de *attributePath* en la entity selection

### [.contains\( entity : 4D.Entity \) : Boolean](#)

devuelve true si la referencia de entidad pertenece a la entity selection

### [.count\( attributePath : Text \) : Real](#)

devuelve el número de entidades en la entity selection con un valor no null en *attributePath*

### [.distinct\( attributePath : Text { ; option : Integer } \) : Collection](#)

devuelve una colección que contiene sólo valores distintos (diferentes) del *attributePath* en la selección de la entidad

### [.drop\( { mode : Integer } \) : 4D.EntitySelection](#)

elimina las entidades pertenecientes a la selección de entidades de la tabla relacionada con su dataclass en el datastore

### [.extract\( attributePath : Text { ; option : Integer } \) : Collection](#)

### [.extract\( attributePath { ; targetPath } { ; ...attributePathN : Text ; targetPathN : Text } \) : Collection](#)

retorna una colección que contiene los valores de *attributePath* extraídos de la entity selection

### [.first\(\) : 4D.Entity](#)

devuelve una referencia a la entidad en la primera posición de la entity selection

### [.getDataClass\(\) : 4D.DataClass](#)

devuelve la dataclass de la entity selection

Devuelve:

<code>.isAlterable() : Boolean</code>	devuelve True si la entity selection es modificable
<code>.isOrdered() : Boolean</code>	devuelve True si la entity selection está ordenada
<code>.last() : 4D.Entity</code>	devuelve una referencia a la entidad en última posición de la entity selection
<code>.length: Integer</code>	devuelve el número de entidades en la entity selection
<code>.max( attributePath : Text ) : any</code>	devuelve el valor más alto (o máximo) entre todos los valores de <i>attributePath</i> en la entity selection
<code>.min( attributePath : Text ) : any</code>	devuelve el valor más bajo (o mínimo) entre todos los valores de <i>attributePath</i> en la entity selection
<code>.minus( entity : 4D.Entity ) : 4D.EntitySelection</code>	
<code>.minus( entitySelection : 4D.EntitySelection ) : 4D.EntitySelection</code>	excluye de la entity selection a la que se aplica la <i>entity</i> o las entidades de <i>entitySelection</i> y devuelve la entity selection resultante
<code>.or( entity : 4D.Entity ) : 4D.EntitySelection</code>	
<code>.or( entitySelection : 4D.EntitySelection ) : 4D.EntitySelection</code>	combina la entity selection con el parámetro <i>entity</i> o <i>entitySelection</i> utilizando el operador lógico (no excluyente) OR
<code>.orderBy( pathString : Text ) : 4D.EntitySelection</code>	
<code>.orderBy( pathObjects : Collection ) : 4D.EntitySelection</code>	devuelve una nueva entity selection ordenada que contiene todas las entidades de la entity selection en el orden especificado por los criterios <i>pathString</i> o <i>pathObjects</i>
<code>.orderByFormula( formulaString : Text { ; sortOrder : Integer } { ; settings : Object} ) : 4D.EntitySelection</code>	
<code>.orderByFormula( formulaObj : Object { ; sortOrder : Integer } { ; settings : Object} ) : 4D.EntitySelection</code>	devuelve una nueva entity selection ordenada
<code>.query( queryString : Text { ; ...value : any } { ; querySettings : Object } ) : 4D.EntitySelection</code>	
<code>.query( formula : Object { ; querySettings : Object } ) : 4D.EntitySelection</code>	busca entidades que cumplan los criterios de búsqueda especificados en <i>queryString</i> o <i>formula</i> y (opcionalmente) <i>value(s)</i> entre todas las entidades de la selección de entidades
<code>.queryPath : Text</code>	contiene una descripción detallada de la búsqueda tal y como fue realizada por 4D
<code>.queryPlan : Text</code>	contiene una descripción detallada de la búsqueda justo antes de su ejecución (es decir, la búsqueda planeada)
<code>.refresh()</code>	invalida inmediatamente los datos de la entity selection en la caché local de ORDA
<code>.selected( selectedEntities : 4D.EntitySelection ) : Object</code>	devuelve un objeto que describe la(s) posición(es) de <i>selectedEntities</i> en la selección de entidades original
<code>.slice( startFrom : Integer { ; end : Integer } ) : 4D.EntitySelection</code>	devuelve una parte de una selección de entidades en una nueva entity selection

`.sum( attributePath : Text ) : Real`

devuelve la suma de todos los valores de `attributePath` en la entity selection

`.toCollection( { options : Integer { ; begin : Integer { ; howMany : Integer } } } ) : Collection`

`.toCollection( filterString : Text {; options : Integer { ; begin : Integer { ; howMany : Integer } } } ) : Collection`

`.toCollection( filterCol : Collection {; options : Integer { ; begin : Integer { ; howMany : Integer } } } ) : Collection`

crea y devuelve una colección donde cada elemento es un objeto que contiene un conjunto de propiedades y valores

## Crear una entity selection

Create entity selection ( `dsTable` : Table { ; `settings` : Object } ) : 4D.EntitySelection

Parámetros	Tipo		Descripción
<code>dsTable</code>	Tabla	->	Tabla de la base 4D cuya selección actual se utilizará para construir la selección de entidades
parámetros	Objeto	->	Opciones de construcción: context
Resultado	4D.EntitySelection	<-	Selección de entidades que coinciden con la clase de datos relacionada con la tabla dada

### Descripción

El comando `Create entity selection` construye y devuelve una nueva, `alterable` selección de entidad relacionada con la clase de datos que coincide con la tabla `dsTable` dada, según la selección actual de esta tabla.

Si la selección actual está ordenada, se crea una selección de entidades `ordenada` (se mantiene el orden de la selección actual). Si la selección actual no está ordenada, se crea una selección de entidades no ordenada.

Si la tabla `dsTable` no está expuesto en `ds`, se devuelve un error. Este comando no puede utilizarse con un datastore remoto.

En el parámetro opcional `settings`, puede pasar un objeto que contenga la siguiente propiedad:

Propiedad	Tipo	Descripción
<code>context</code>	Texto	Etiqueta para el <code>contexto de optimización</code> aplicado a la entity selection.

### Ejemplo

```
var $employees : cs.EmployeeSelection
ALL RECORDS([Employee])
$employees:=Create entity selection([Employee])
// La entity selection $employees ahora contiene un conjunto de referencias
// en todas las entidades relacionadas con la clase de datos Employee
```

### Ver también

`dataClass.newSelection()`

## USE ENTITY SELECTION

USE ENTITY SELECTION (`entitySelection`)

Parámetros	Tipo		Descripción
entitySelection	EntitySelection	->	Una entity selection

## Descripción

El comando `USE ENTITY SELECTION` actualiza la selección actual de la tabla que coincide con la dataclass del parámetro `entitySelection`, según el contenido de la entity selection.

Este comando no puede utilizarse con un [datastore remoto](#).

Después de una llamada a `USE ENTITY SELECTION`, el primer registro de la selección actual actualizada (si no está vacío) se convierte en el registro actual, pero no se carga en la memoria. Si necesita utilizar los valores de los campos del registro actual, utilice el comando `LOAD RECORD` después del comando `USE ENTITY SELECTION`.

## Ejemplo

```
var $entitySel : Object

$entitySel:=ds.Employee.query("lastName = :1;"M@") // $entitySel está asociado a la dataclass Employee
REDUCE SELECTION([Employee];0)
USE ENTITY SELECTION($entitySel) // Se actualiza la selección actual de la tabla Employee
```

## [index]

► Histórico

`[index]` : 4D.Entity

## Descripción

La notación `EntitySelection[index]` permite acceder a las entidades dentro de la selección de entidades utilizando la sintaxis de colección estándar: pase la posición de la entidad que desea obtener en el parámetro `index`.

Tenga en cuenta que la entidad correspondiente se vuelve a cargar desde el almacén de datos.

`index` puede ser cualquier número entre 0 y `.length -1`.

- Si `index` está fuera de rango, se devuelve un error.
- Si `index` corresponde a una entidad descartada, se devuelve un valor Null.

Atención: `EntitySelection[index]` es una expresión no assignable, lo que significa que no puede utilizarse como referencia editable de la entidad con métodos como `.lock()` o `.save()`. Para trabajar con la entidad correspondiente, es necesario asignar la expresión devuelta a una expresión assignable, como una variable. Ejemplos:

```
$sel:=ds.Employee.all() //creación de la entity selection
//declaraciones no válidas:
$result:=$sel[0].lock() //NO funcionará
$sel[0].lastName:="Smith" //NO funcionará
$result:=$sel[0].save() //NO funcionará
//valid code:
$entity:=$sel[0] //OK
$entity.lastName:="Smith" //OK
$entity.save() //OK
```

## Ejemplo

```
var $employees : cs.EmployeeSelection
var $employee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName = :1";"H@")
$employee:=$employees[2] // La tercera entidad de la selección de entidades $employees se recarga de l
```

## .attributeName

► Histórico

.attributeName : Collection  
.attributeName : 4D.EntitySelection

### Descripción

Todo atributo de dataclass puede ser utilizado como una propiedad de una entity selection para devolver una "proyección" de los valores del atributo en la entity selection. Los valores proyectados pueden ser una colección o una nueva selección de entidades, dependiendo de `kind` (`storage` o `relation`) del atributo.

- Si el "kind" de `attributeName` es `storage` : `.attributeName` devuelve una colección de valores del mismo tipo que `attributeName`.
- Si el "kind" de `attributeName` es `relatedEntity` : `.attributeName` devuelve una nueva entity selection de valores relacionados del mismo tipo que `attributeName`. Se eliminan los duplicados (se devuelve una entity selection desordenada).
- Si el "kind" de `attributeName` es `relatedEntities` : `.attributeName` devuelve una nueva entity selection de valores relacionados del mismo tipo que `attributeName`. Se eliminan los duplicados (se devuelve una entity selection desordenada).

Cuando se utiliza un atributo de relación como propiedad de una selección de entidades, el resultado es siempre otra selección de entidades, aunque sólo se devuelva una entidad. Cuando se utiliza un atributo de relación como propiedad de una selección de entidades, el resultado es siempre otra selección de entidades, aunque sólo se devuelva una entidad.

Si el atributo no existe en la selección de entidades, se devuelve un error.

## Ejemplo 1

Proyección de valores de almacenamiento:

```
var $firstNames : Collection
$entitySelection:=ds.Employee.all()
$firstNames:=$entitySelection.firstName // firstName es un string
```

La colección resultante es una colección de cadenas, por ejemplo:

```
[  
  "Joanna",  
  "Alexandra",  
  "Rick"  
]
```

## Ejemplo 2

Proyección de la entidad relacionada:

```

var $es; $entitySelection : cs.EmployeeSelection
$entitySelection:=ds.Employee.all()
$es:=$entitySelection.employer // employer está relacionado a la dataClass Company

```

El objeto resultante es una selección de entidades de la empresa con los duplicados eliminados (si los hay).

### Ejemplo 3

Proyección de entidades relacionadas:

```

var $es : cs.EmployeeSelection
$es:=ds.Employee.all().directReports // directReports está relacionado a la dataclass Employee

```

El objeto resultante es una entity selection de la dataclass Employee sin duplicados (si los hay).

## .add()

► Histórico

`.add( entity : 4D.Entity ) : 4D.EntitySelection`

Parámetros	Tipo		Descripción
entity	4D.Entity	->	Entidad que debe añadirse a la entity selection
Resultado	4D.EntitySelection	->	Selección de entidades incluyendo la <code>entity</code> añadida

### Descripción

La función `.add()` añade la `entity` especificada a la entity selection y devuelve la selección de entidades modificada.

Los valores de tipo Date se convierten en valores numéricos (segundos) y se utilizan para calcular la media.

Atención: la entity selection debe ser *alterable*, es decir, ha sido creado, por ejemplo, por `.newSelection()` o `Create entity selection`, de lo contrario `.add()` devolverá un error. Las entity selections compatibles no aceptan la adición de entidades. Para más información, consulte la sección [Entity selections compatibles o modificables](#) section.

- Si la entity selection está ordenada, `entity` se añade al final de la selección. Si una referencia a la misma entidad ya pertenece a la selección de entidades, se duplica y se añade una nueva referencia.
- Si la entity selection no está ordenada, `entity` se añade en cualquier lugar de la selección, sin un orden específico.

Para más información, consulte la sección [Entity selections ordenada o no ordenadas](#).

La entity selection modificada es devuelta por la función, de modo que las llamadas a la función pueden encadenarse.

Se produce un error si `entity` y la entity selection no están relacionadas con la misma dataclass. Si `entity` es Null, no se produce ningún error.

### Ejemplo 1

```

var $employees : cs.EmployeeSelection
var $employee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName = :1";"S@")
$employee:=ds.Employee.new()
$employee.lastName:="Smith"
$employee.save()
$employees.add($employee) //La entidad $employee se añade a la entity selection $employees

```

## Ejemplo 2

Las llamadas a la función se pueden encadenar:

```
var $sel : cs.ProductSelection
var $p1;$p2;$p3 : cs.ProductEntity

$p1:=ds.Product.get(10)
$p2:=ds.Product.get(11)
$p3:=ds.Product.get(12)
$sel:=ds.Product.query("ID > 50")
$sel:=$sel.add($p1).add($p2).add($p3)
```

## .and()

► Histórico

.and(*entity* : 4D.Entity) : 4D.EntitySelection  
.and(*entitySelection* : 4D.EntitySelection) : 4D.EntitySelection

Parámetros	Tipo		Descripción
entity	4D.Entity	->	Entidad a intersectar
entitySelection	4D.EntitySelection	->	Entity selection a intersectar
Resultado	4D.EntitySelection	<-	Entity selection resultante de la intersección con el operador lógico AND

### Descripción

La función `.and()` combina la entity selection con el parámetro *entity* o *entitySelection* utilizando el operador lógico AND; devuelve una nueva entity selection no ordenada que contiene sólo las entidades a las que se hace referencia tanto en la entity selection y el parámetro.

- Si pasa *entity* como parámetro, se combina esta entidad con la entity selection. Si la entidad pertenece a la entity selection, se devuelve una nueva entity selection que sólo contiene la entidad. En caso contrario, se devuelve una selección de entidades vacía.
- Si pasa *entitySelection* como parámetro, combina ambas entity selections. Se devuelve una nueva selección de entidades que contiene sólo las entidades a las que se hace referencia en ambas selecciones. Si no hay ninguna entidad intersectada, se devuelve una entity selection vacía.

Puede comparar [entity selections ordenadas y/o desordenadas](#). La selección resultante es siempre desordenada.

Si la entity selection inicial o el parámetro *entitySelection* están vacíos, o si *entity* es Null, se devuelve una entity selection vacía.

Si la entity selection inicial y el parámetro no están relacionados con la misma dataclass, se produce un error.

## Ejemplo 1

```

var $employees; $result : cs.EmployeeSelection
var $employee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName = :1";"H@")
//la entity selection $employees contiene la entidad
//con la llave primaria 710 y otras entidades
//por ejemplo. "Colin Hetrick" / "Grady Harness" / "Sherlock Holmes" (primary key 710)
$employee:=ds.Employee.get(710) // Devuelve "Sherlock Holmes"

$result:=$employees.and($employee) // $result es una selección de entidades que contiene
//sólo la entidad con llave primaria 710 ("Sherlock Holmes")

```

## Ejemplo 2

Queremos tener una selección de empleados llamados "Jones" que vivan en Nueva York:

```

var $sel1; $sel2; $sel3 : cs.EmployeeSelection
$sel1:=ds.Employee.query("name =:1";"Jones")
$sel2:=ds.Employee.query("city=:1";"New York")
$sel3:=$sel1.and($sel2)

```

## .average()

► Histórico

`.average( attributePath : Text ) : Real`

Parámetros	Tipo		Descripción
attributePath	Texto	->	Ruta del atributo a utilizar para el cálculo
Resultado	Real	<-	Media aritmética (promedio) de los valores de las entidades para el atributo (No se define para una entity selection vacía)

### Descripción

La función `.average()` devuelve la media aritmética (promedio) de todos los valores no nulos de `attributePath` en la entity selection.

Pase en el parámetro `attributePath` la ruta del atributo a evaluar.

Sólo se tienen en cuenta los valores numéricos para el cálculo. Tenga en cuenta, sin embargo, que si el `attributePath` de la selección de entidades contiene tipos de valores mixtos, `.average()` tiene en cuenta todos los elementos escalares para calcular el valor medio.

Para más información sobre propiedad compatible de entity selections, consulte la sección [Entity selections compatibles o modificables](#).

`.average()` devuelve `undefined` si la entity selection está vacía o `attributePath` no contiene valores numéricos.

Se devuelve un error si:

- `>attributePath` es un atributo relativo,
- `attributePath` designa un atributo que no existe en la dataclass de la entity selection.

## Ejemplo

Queremos obtener una lista de empleados cuyo salario es superior al salario medio:

```

var $averageSalary : Real
var $moreThanAv : cs.EmployeeSelection
$averageSalary:=ds.Employee.all().average("salary")
$moreThanAv:=ds.Employee.query("salary > :1";$averageSalary)

```

## .contains()

► Histórico

`.contains( entity : 4D.Entity ) : Boolean`

Parámetros	Tipo		Descripción
entity	4D.Entity	->	Entidad a evaluar
Resultado	Booleano	<-	True si la entidad pertenece a la entity selection, de lo contrario False

### Descripción

La función `.contains()` devuelve true si la referencia de entidad pertenece a la entity selection, y false en caso contrario.

En `entity`, especifique la entidad a buscar en la entity selection. Si la entidad es Null, la función devolverá false.

Si `entity` y la entity selection no pertenecen a la misma dataclass, se produce un error.

### Ejemplo

```

var $employees : cs.EmployeeSelection
var $employee : cs.EmployeeEntity

$employees:=ds.Employee.query("lastName=:1";"H@")
$employee:=ds.Employee.get(610)

If($employees.contains($employee))
    ALERT("La entidad con llave primaria 610 tiene un apellido que empieza por H")
Else
    ALERT("La entidad con llave primaria 610 no tiene un apellido que empiece por H")
End if

```

## .count()

► Histórico

`.count( attributePath : Text ) : Real`

Parámetros	Tipo		Descripción
attributePath	Texto	->	Ruta del atributo que se utilizará para el cálculo
Resultado	Real	<-	Número de valores de <code>attributePath</code> no null en la entity selection

### Descripción

La función `.count()` devuelve el número de entidades en la entity selection con un valor no null en `attributePath`.

Sólo se tienen en cuenta los valores escalares. Los valores de tipo objeto o colección se consideran valores nulos.

Se devuelve un error si:

- `>attributePath` es un atributo relativo,
- `attributePath` no se encuentra en la clase de datos de la entity selection.

## Ejemplo

Queremos averiguar el número total de empleados de una empresa sin contar a los que no se les ha especificado el cargo:

```
var $sel : cs.EmployeeSelection
var $count : Real

$sel:=ds.Employee.query("employer = :1";"Acme, Inc")
$count:=$sel.count("jobtitle")
```

## .copy()

► Histórico

`.copy( { option : Integer } ) : 4D.EntitySelection`

Parámetros	Tipo		Descripción
option	Integer	->	ck shared : devuelve una entity selection compatible
Resultado	4D.EntitySelection	<-	Copia de la entity selection

### Descripción

La función `.copy()` devuelve una copia de la entity selection original.

Las entidades de una colección de entidades a las que se accede por medio de [ ] no se recargan desde la base de datos.

Por defecto, si se omite el parámetro `option`, la función devuelve una nueva entity selection alterable (incluso si la función se aplica a una entity selection compatible). Pasa la constante `ck shared` en el parámetro `option` si quiere crear una entity selection compatible.

Esta función siempre devuelve True cuando la entity selection proviene de un datastore remoto.

## Ejemplo

Se crea una nueva entidad vacía de selección de productos cuando se carga el formulario:

```
Case of
  :(Form event code=On Load)
    Form.products:=ds.Products.newSelection()
End case
```

A continuación, esta selección de entidades se actualiza con productos y se desea compartir los productos entre varios procesos. Se copia la selección de la entidad `Form.products` como compatible:

```

...
// La selección de entidades de Form.products se actualiza
Form.products.add(Form.selectedProduct)

Use(Storage)
  If(Storage.products=Null)
    Storage.products:=New shared object()
  End if

  Use(Storage.products)
    Storage.products:=Form.products.copy(ck shared)
  End use
End use

```

## .distinct()

► Histórico

`.distinct( attributePath : Text { ; option : Integer } ) : Collection`

Parámetros	Tipo		Descripción
attributePath	Texto	->	Ruta del atributo cuyos valores distintos desea obtener
option	Integer	->	<code>dk diacrítico</code> : evaluación diacrítica ("A" # "a" por ejemplo)
Resultado	Collection	<-	Colección con sólo valores distintos

### Descripción

La función `.distinct()` devuelve una colección que contiene sólo valores distintos (diferentes) del `attributePath` en la selección de la entidad.

La colección devuelta se clasifica automáticamente. Los valores Null no se devuelven.

En el parámetro `attributePath`, pase el atributo de entidad cuyos valores distintos quiere obtener. Sólo se pueden manejar valores escalares (texto, número, booleano o fecha). Los tipos se devuelven en el siguiente orden: Si `attributePath` lleva a una propiedad de objeto que contiene valores de diferentes tipos, primero se agrupan por tipo y se ordenan después.

1. booleanos
2. strings
3. numbers
4. fechas

Se puede utilizar la notación `[]` para designar una colección cuando `attributePath` es una ruta dentro de un objeto (ver ejemplos).

Por defecto, se realiza una evaluación no diacrítica. Si desea que la evaluación distinga entre mayúsculas y minúsculas o que diferencie los caracteres acentuados, pase la constante `dk diacritical` en el parámetro `option`.

Se devuelve un error si:

- `>attributePath` es un atributo relativo,
- `attributePath` no se encuentra en la clase de datos de la entity selection.

### Ejemplos

Quiere obtener una colección que contenga un solo elemento por nombre de país:

```

var $countries : Collection
$countries:=ds.Employee.all().distinct("address.country")

```

`nicknames` es una colección y `extra` es un atributo de objeto:

```
$values:=ds.Employee.all().distinct("extra.nicknames[].first")
```

## .drop()

► Histórico

`.drop( { mode : Integer } ) : 4D.EntitySelection`

Parámetros	Tipo		Descripción
mode	Integer	->	<code>dk stop dropping on first error</code> : detiene la ejecución del método en la primera entidad no suprimible
Resultado	4D.EntitySelection	<-	Entity selection vacía si se ejecuta con éxito, si no entity selection que contengan las entidades no eliminables

### Descripción

La función `.drop()` elimina las entidades pertenecientes a la selección de entidades de la tabla relacionada con su dataclass en el datastore. La entity selection permanece en la memoria.

La eliminación de entidades es permanente y no se puede deshacer. Se recomienda llamar a esta acción en una transacción para tener una opción de recuperación.

Si se encuentra una entidad bloqueada durante la ejecución de `.drop()`, no se elimina. Por defecto, el método procesa todas las entidades de la selección de entidades y devuelve las entidades no eliminables en la selección de entidades. Si quiere que el método detenga la ejecución en la primera entidad no suprimible encontrada, pase la constante `dk stop dropping on first error` en el parámetro `mode`.

### Ejemplo

Ejemplo sin la opción `dk stop dropping on first error`:

```
var $employees; $notDropped : cs.EmployeeSelection
$employees:=ds.Employee.query("firstName=:1";"S@")
$notDropped:=$employees.drop() // $notDropped es una entity selection que contiene todas las entidades
If($notDropped.length=0) //La acción de eliminación es exitosa, todas las entidades han sido eliminadas
    ALERT("You have dropped "+String($employees.length)+" employees") //La selección de entidades elimin
Else
    ALERT("Problem during drop, try later")
End if
```

Ejemplo con la opción `dk stop dropping on first error`:

```
var $employees; $notDropped : cs.EmployeeSelection
$employees:=ds.Employee.query("firstName=:1";"S@")
$notDropped:=$employees.drop(dk stop dropping on first error) // $notDropped es una entity selection que
If($notDropped.length=0) //La acción de eliminación es exitosa, todas las entidades han sido eliminadas
    ALERT("You have dropped "+String($employees.length)+" employees") //La selección de entidades elimin
Else
    ALERT("Problem during drop, try later")
End if
```

## .extract()

► Histórico

.extract( *attributePath* : Text { ; *option* : Integer } ) : Collection

.extract( *attributePath* { ; *targetPath* } { ; ...*attributePathN* : Text ; *targetPathN* : Text } ) : Collection

Parámetros	Tipo		Descripción
<i>attributePath</i>	Texto	->	Ruta del atributo cuyos valores deben ser extraídos en la nueva colección
<i>targetPath</i>	Texto	->	Ruta o nombre del atributo objetivo
<i>option</i>	Integer	->	<code>ck keep null</code> : incluye los atributos null en la colección devuelta (ignorados por defecto)
Resultado	Collection	<-	Colección que contiene los valores extraídos

### Descripción

La función `.extract()` retorna una colección que contiene los valores de *attributePath* extraídos de la entity selection.

*attributePath* puede referirse a:

- un atributo escalar de dataclass,
- entidad relacionada,
- entidades relacionadas.

Si *attributePath* no es válido, se devuelve una colección vacía.

Esta función acepta dos sintaxis.

`.extract( attributePath : Text { ; option : Integer } ) : Collection`

Con esta sintaxis, `.extract()` llena la colección devuelta con los valores *attributePath* de la entity selection.

Por defecto, las entidades para las que *attributePath* es *null* o indefinida se ignoran en la colección resultante. Puede pasar la constante `ck keep null` en el parámetro *option* para incluir estos valores como elementos *null* en la colección devuelta.

- Los atributos dataclass `.kind = "relatedEntity"` se extraen como una colección de entidades (se mantienen las duplicaciones).
- Los atributos dataclass con `.kind = "relatedEntities"` se extraen como una colección de entity selections.

`.extract( attributePath ; targetPath { ; ...attributePathN ; ... targetPathN} ) : Collection`

Con esta sintaxis, `.extract()` llena la colección devuelta con las propiedades *attributePath*. Cada elemento de la colección devuelta es un objeto con las propiedades *targetPath* llenadas con las propiedades *attributePath* correspondientes. Se mantienen los valores *null* (el parámetro *option* se ignora) con esta sintaxis.

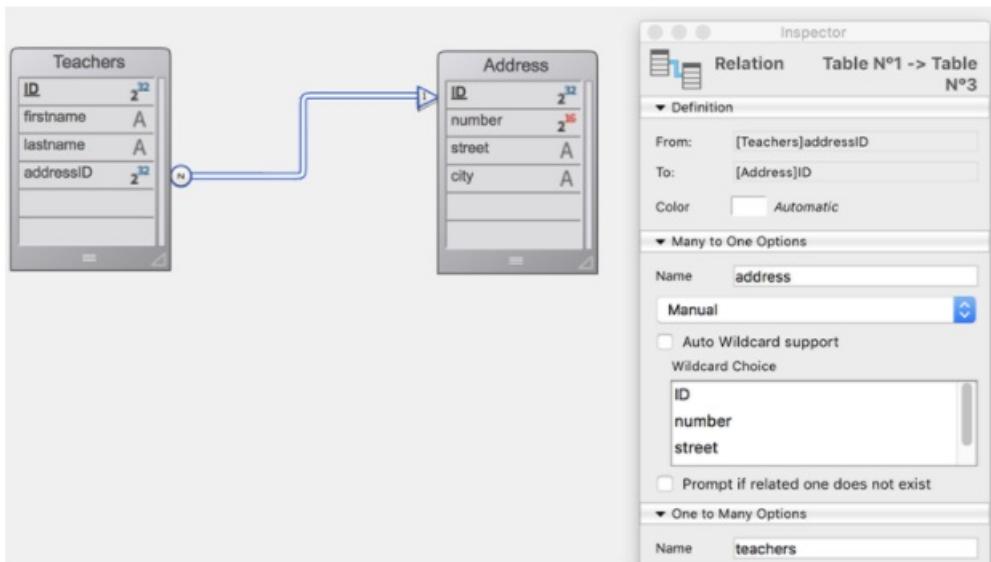
Si se dan varios *attributePath*, debe darse un *targetPath* para cada uno. Sólo se extraen los pares válidos [ *attributePath*, *targetPath* ].

- Los atributos dataclass con `.kind = "relatedEntity"` se extraen como una entidad.
- Los atributos dataclass con `.kind = "relatedEntities"` se extraen como una entity selection.

Los valores Null se evalúan como inferiores a los otros valores.

### Ejemplo

Dada la siguiente tabla y relación:



```

var $firstnames; $addresses; $mailing; $teachers : Collection
//
// $firstnames es una colección de cadenas
$firstnames:=ds.Teachers.all().extract("firstname")
//
// $addresses es una colección de entidades relacionadas con la dataclass Address
// Se extraen los valores null de Address
$addresses:=ds.Teachers.all().extract("address";ck keep null)
//
// $mailing es una colección de objetos con las propiedades "who" y "to"
// El contenido de la propiedad "who" es de tipo Cadena
// El contenido de la propiedad "to" es de tipo entity (dataclass Address)
$mailing:=ds.Teachers.all().extract("lastname";"who";"address";"to")
//
// $mailing es una colección de objetos con las propiedades "who" y "city"
// El contenido de la propiedad "who" es de tipo Cadena
// El contenido de la propiedad "city" es de tipo Cadena
$mailing:=ds.Teachers.all().extract("lastname";"who";"address.city";"city")
//
// $teachers es una colección de objetos con las propiedades "where" y "who"
// El contenido de la propiedad "where" es de tipo Cadena
// El contenido de la propiedad "who" es una entity selection (dataclass Teachers)
$teachers:=ds.Address.all().extract("city";"where";"teachers";"who")
//
// $teachers es una colección de entity selections
$teachers:=ds.Address.all().extract("teachers")

```

## .first()

► Histórico

.first() : 4D.Entity

Parámetros	Tipo		Descripción
Resultado	4D.Entity	<-	Referencia a la primera entidad de la entity selection (Null si la selección está vacía)

### Descripción

La función `.first()` devuelve una referencia a la entidad en la primera posición de la entity selection.

El resultado de esta función es similar a:

```
$entity:=$entitySel[0]
```

Sin embargo, hay una diferencia entre ambas afirmaciones cuando la selección está vacía:

```
var $entitySel : cs.EmpSelection
var $entity : cs.EmpEntity
$entitySel:=ds.Emp.query("lastName = :1";"Nonexistentname") //ninguna entidad correspondiente
//entity selection is then empty
$entity:=$entitySel.first() //devuelve Null
$entity:=$entitySel[0] //genera un error
```

## Ejemplo

```
var $entitySelection : cs.EmpSelection
var $entity : cs.EmpEntity
$entitySelection:=ds.Emp.query("salary > :1";100000)
If($entitySelection.length#0)
    $entity:=$entitySelection.first()
End if
```

## .getDataClass()

► Histórico

.getDataClass() : 4D.DataClass

Parámetros	Tipo		Descripción
Resultado	4D.DataClass	<-	DataClass a la que pertenece la entity selection

## Descripción

La función `.getDataClass()` devuelve la dataclass de la entity selection.

Esta función es principalmente útil en el contexto del código genérico.

## Ejemplo

El siguiente código genérico duplica todas las entidades de la entity selection:

```
//método duplicate_entities
//duplicate_entities($entity_selection)

#DECLARE ( $entitySelection : 4D.EntitySelection )
var $dataClass : 4D.DataClass
var $entity; $duplicate : 4D.Entity
var $status : Object
$dataClass:=$entitySelection.getDataClass()
For each($entity;$entitySelection)
    $duplicate:=$dataClass.new()
    $duplicate.fromObject($entity.toObject())
    $duplicate[$dataClass.getInfo().primaryKey]:=Null //restablecer la llave primaria
    $status:=$duplicate.save()
End for each
```

## .getRemoteContextAttributes()

► Histórico

Devuelve:

Parámetros	Tipo		Descripción
result	Texto	<-	Atributos de contexto vinculados a la entity selection, separados por una coma

Modo avanzado: esta función está pensada para los desarrolladores que necesitan personalizar las funcionalidades por defecto de ORDA para configuraciones específicas. En la mayoría de los casos, no será necesario utilizarla.

### Descripción

La función `.getRemoteContextAttributes()` devuelve información sobre el contexto de optimización utilizado por la entity selection.

Si no hay [contexto de optimización](#) para la entity selection, la función devuelve un texto vacío.

### Ejemplo

```
var $ds : 4D.DataStoreImplementation
var $persons : cs.PersonsSelection
var $p : cs.PersonsEntity

var $info : Text
var $text : Text

$ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")

$persons:=$ds.Persons.all()
$text:=""
For each ($p; $persons)
    $text:=$p.firstname+" lives in "+$p.address.city+ " "
End for each

$info:=$persons.getRemoteContextAttributes()
//$info = "firstname,address,address.city"
```

### Ver también

[Entity.getRemoteContextAttributes\(\)](#)  
[.clearAllRemoteContexts\(\)](#)  
[.getRemoteContextInfo\(\)](#)  
[.getAllRemoteContexts\(\)](#)  
[.setRemoteContextInfo\(\)](#)

## .isAlterable()

► Histórico

`.isAlterable() : Boolean`

Parámetros	Tipo		Descripción
Resultado	Booleano	<-	True si la entity selection es modificable, de lo contrario False

### Descripción

La función `.isAlterable()` devuelve True si la entity selection es modificable, y False si no lo es.

Para más información, consulte la sección [Entity selections compatibles o modificables](#).

## Ejemplo

Va a mostrar `Form.products` en un [list box](#) para que el usuario pueda añadir nuevos productos. Quiere asegurarse de que es modificable para que el usuario pueda añadir nuevos productos sin error:

```
If (Not(Form.products.isAlterable()))
    Form.products:=Form.products.copy()
End if
...
Form.products.add(Form.product)
```

## .isOrdered()

► Histórico

`.isOrdered() : Boolean`

Parámetros	Tipo		Descripción
Resultado	Booleano	<-	True si la entity selection es ordenada, de lo contrario False

### Descripción

La función `.isOrdered()` devuelve True si la entity selection está ordenada, y False si está desordenada.

Esta función no modifica la selección de entidades original.

Para más información, consulte [Entity selection ordenadas o desordenadas](#).

## Ejemplo

```
var $employees : cs.EmployeeSelection
var $employee : cs.EmployeeEntity
var $isOrdered : Boolean
$employees:=ds.Employee.newSelection(dk keep ordered)
$employee:=ds.Employee.get(714) // Obtiene la entidad con llave primaria 714

//En una entity selection ordenada, podemos añadir la misma entidad varias veces (los duplicados se mantienen)
$employees.add($employee)
$employees.add($employee)
$employees.add($employee)

$isOrdered:=$employees.isOrdered()
If($isOrdered)
    ALERT("The entity selection is ordered and contains "+String($employees.length)+" employees")
End if
```

## .last()

► Histórico

`.last() : 4D.Entity`

Parámetros	Tipo		Descripción
Resultado	4D.Entity	<-	Referencia a la última entidad de la entity selection (Null si la entity selection está vacía)

## Descripción

La función `.last()` devuelve una referencia a la entidad en última posición de la entity selection.

El resultado de esta función es similar a:

```
$entity:=$entitySel[length-1]
```

Si la entity selection está vacía, la función devuelve Null.

## Ejemplo

```
var $entitySelection : cs.EmpSelection
var $entity : cs.EmpEntity
$entitySelection:=ds.Emp.query("salary < :1";50000)
If($entitySelection.length#0)
    $entity:=$entitySelection.last()
End if
```

## .length

► Histórico

.length: Integer

## Descripción

La propiedad `.length` devuelve el número de entidades en la entity selection. Si la entity selection está vacía, devuelve 0.

Las entity selections siempre tienen una propiedad `.length`.

Las entidades de una colección de entidades a las que se accede por medio de [ ] no se recargan desde la base de datos.

## Ejemplo

```
var $vSize : Integer
$vSize:=ds.Employee.query("gender = :1";"male").length
ALERT(String(vSize)+" male employees found.")
```

## .max()

► Histórico

.max( *attributePath* : Text ) : any

Parámetros	Tipo		Descripción
attributePath	Texto	->	Ruta del atributo que se utilizará para el cálculo
Resultado	any	<-	Valor más alto del atributo

## Descripción

La función `.max()` devuelve el valor más alto (o máximo) entre todos los valores de `attributePath` en la entity selection. En realidad devuelve el valor de la última entidad de la selección de entidades tal y como se ordenaría de forma ascendente utilizando la función `.orderBy()`.

Si se pasa en `attributePath` una ruta a una propiedad de objeto que contiene diferentes tipos de valores, la función `.max()` devolverá el valor máximo dentro del primer tipo escalar en el orden de la lista de tipos 4D por defecto (ver la descripción de `.sort()`).

`.max()` devuelve undefined si la entity selection está vacía o si `attributePath` no se encuentra en el atributo objeto.

Se devuelve un error si:

- `>attributePath` es un atributo relativo,
- `attributePath` designa un atributo que no existe en la dataclass de la entity selection.

## Ejemplo

Queremos encontrar el salario más alto entre todas las empleadas:

```
var $sel : cs.EmpSelection
var $maxSalary : Real
$sel:=ds.Employee.query("gender = :1";"female")
$maxSalary:=$sel.max("salary")
```

## .min()

► Histórico

`.min( attributePath : Text ) : any`

Parámetros	Tipo		Descripción
attributePath	Texto	->	Ruta del atributo que se utilizará para el cálculo
Resultado	any	<-	Valor más bajo del atributo

## Descripción

La función `.min()` devuelve el valor más bajo (o mínimo) entre todos los valores de `attributePath` en la entity selection. En realidad devuelve el valor de la primera entidad de la selección si se ordenara de forma ascendente utilizando la función `.orderBy()` (excluyendo los valores null).

Si se pasa en `attributePath` una ruta a una propiedad objeto que contiene diferentes tipos de valores, la función `.min()` devolverá el valor mínimo dentro del primer tipo de valor escalar en el orden de la lista de tipos (ver la descripción de `.sort()`).

`.min()` devuelve undefined si la entity selection está vacía o si `attributePath` no se encuentra en el atributo objeto.

Se devuelve un error si:

- `>attributePath` es un atributo relativo,
- `attributePath` designa un atributo que no existe en la dataclass de la entity selection.

## Ejemplo

En este ejemplo, queremos encontrar el salario más bajo entre todas las empleadas:

```

var $sel : cs.EmpSelection
var $minSalary : Real
$sel:=ds.Employee.query("gender = :1";"female")
$minSalary:=$sel.min("salary")

```

## .minus()

► Histórico

.minus( *entity* : 4D.Entity ) : 4D.EntitySelection  
 .minus( *entitySelection* : 4D.EntitySelection ) : 4D.EntitySelection

Parámetros	Tipo		Descripción
entity	4D.Entity	->	Entidad a sustraer
entitySelection	4D.EntitySelection	->	Entity selection a sustraer
Resultado	4D.EntitySelection	<-	Nueva entity selection o una nueva referencia en la entity selection existente

### Descripción

La función `.minus()` excluye de la entity selection a la que se aplica la *entity* o las entidades de *entitySelection* y devuelve la entity selection resultante.

- Si se pasa *entity* como parámetro, la función crea una nueva entity selection sin *entity* (si *entity* pertenece a la entity selection). Si *entity* no estaba incluida en la entity selection original, se devuelve una nueva referencia a la entity selection.
- Si se pasa *entitySelection* como parámetro, la función devuelve una entity selection que contiene las entidades pertenecientes a la entity selection original sin las entidades pertenecientes a *entitySelection*.

Puede comparar [entity selections ordenadas y/o desordenadas](#). La selección resultante es siempre desordenada.

Si la entity selection inicial o la entity selection inicial y la del parámetro *entitySelection* están vacías, se devuelve una entity selection vacía.

Si *entitySelection* está vacía o si *entity* es Null, se devuelve una nueva referencia a la entity selection original.

Si la entity selection inicial y el parámetro no están relacionados con la misma dataclass, se produce un error.

### Ejemplo 1

```

var $employees; $result : cs.EmployeeSelection
var $employee : cs.EmployeeEntity

$employees:=ds.Employee.query("lastName = :1";"H@")
// la entity selection $employees contiene la entidad con llave primaria 710 y otras entidades
// por ejemplo. "Colin Hetrick", "Grady Harness", "Sherlock Holmes" (primary key 710)

$employee:=ds.Employee.get(710) // Devuelve "Sherlock Holmes"

$result:=$employees.minus($employee) // $result contiene "Colin Hetrick", "Grady Harness"

```

### Ejemplo 2

Queremos tener una selección de empleadas llamadas "Jones" que viven en Nueva York :

```

var $sel1; $sel2; $sel3 : cs.EmployeeSelection
$sel1:=ds.Employee.query("name =:1";"Jones")
$sel2:=ds.Employee.query("city=:1";"New York")
$sel3:=$sel1.and($sel2).minus(ds.Employee.query("gender='male'"))

```

## .or()

► Histórico

.or( *entity* : 4D.Entity ) : 4D.EntitySelection  
 .or( *entitySelection* : 4D.EntitySelection ) : 4D.EntitySelection

Parámetros	Tipo		Descripción
entity	4D.Entity	->	Entidad a intersectar
entitySelection	4D.EntitySelection	->	Entity selection a intersectar
Resultado	4D.EntitySelection	<-	Nueva entity selection o una nueva referencia a la entity selection de origen

### Descripción

La función `.or()` combina la entity selection con el parámetro *entity* o *entitySelection* utilizando el operador lógico (no excluyente) OR; devuelve una nueva entitySelection no ordenada que contiene todas las entidades de la entitySelection y del parámetro.

- Si pasa *entity* como parámetro, se compara esta entidad con la entity selection. Si la entidad pertenece a la entity selection, se devuelve una nueva referencia a la entity selection. En caso contrario, se devuelve una nueva entity selection que contiene la entity selection original y la entidad.
- Si pasa *entitySelection* como parámetro, compara ambas entity selections. Se devuelve una nueva entity selection que contiene las entidades pertenecientes a la selección de entidades original o *entitySelection* (o no es exclusiva, las entidades referenciadas en ambas selecciones no se duplican en la selección resultante).

Puede comparar [entity selections ordenadas y/o desordenadas](#). La selección resultante es siempre desordenada.

Si la entity selection inicial y la del parámetro *entitySelection* están vacías, se devuelve una entity selection vacía. Si la entity selection original está vacía, se devuelve una referencia a *entitySelection* o una entity selection que sólo contiene *entity*.

Si *entitySelection* está vacía o si *entity* es Null, se devuelve una nueva referencia a la entity selection original.

Si la entity selection inicial y el parámetro no están relacionados con la misma dataclass, se produce un error.

### Ejemplo 1

```

var $employees1; $employees2; $result : cs.EmployeeSelection
$employees1:=ds.Employee.query("lastName = :1";"Hq") //Devuelve "Colin Hetrick", "Grady Harness"
$employees2:=ds.Employee.query("firstName = :1";"C@") //Devuelve "Colin Hetrick", "Cath Kidston"
$result:=$employees1.or($employees2) // $result contiene "Colin Hetrick", "Grady Harness", "Cath Kidston"

```

### Ejemplo 2

```

var $employees; $result : cs.EmployeeSelection
var $employee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName = :1";"H@") // Devuelve "Colin Hetrick", "Grady Harness", "Sherlock Holmes"
$employee:=ds.Employee.get(686) //la entidad con llave primaria 686 no pertenece a la entity selection
//Coincide con la empleada "Mary Smith"

$result:=$employees.or($employee) //result contiene "Colin Hetrick", "Grady Harness", "Sherlock Holmes"

```

## .orderBy()

► Histórico

.orderBy( *pathString* : Text ) : 4D.EntitySelection  
 .orderBy( *pathObjects* : Collection ) : 4D.EntitySelection

Parámetros	Tipo		Descripción
<i>pathString</i>	Texto	->	Ruta(s) de atributos e instrucciones de clasificación para la entity selection
<i>pathObjects</i>	Collection	->	Colección de objetos criterio
Resultado	4D.EntitySelection	<-	Nueva entity selection en el orden especificado

### Descripción

La función `.orderBy()` devuelve una nueva entity selection ordenada que contiene todas las entidades de la entity selection en el orden especificado por los criterios *pathString* o *pathObjects*.

- Este método no modifica la selección de entidades original.
- Para más información, consulte la sección [Entity selections ordenada o no ordenadas](#).

Debe utilizar un parámetro de criterio para definir cómo deben ordenarse las entidades. Se soportan dos parámetros diferentes:

- *pathString* (Texto): este parámetro contiene una fórmula compuesta de rutas de atributo de 1 a x y (opcionalmente) órdenes de clasificación, separados por comas. La sintaxis es:

```
"attributePath1 {desc or asc}, attributePath2 {desc or asc},..."
```

El orden en que se pasan los atributos determina la prioridad de ordenación de las entidades. Por defecto, los atributos se clasifican en orden ascendente. Puede definir el orden de clasificación de una propiedad en la cadena de criterios, separado de la ruta de la propiedad por un solo espacio: pase "asc" para ordenar en orden ascendente o "desc" en orden descendente.

- *pathObjects* (collection): cada elemento de la colección contiene un objeto estructurado de la siguiente manera:

```
{
  "propertyPath": string,
  "descending": boolean
}
```

Por defecto, los atributos se clasifican en orden ascendente ("descending" es false).

Puede añadir tantos objetos en la colección de criterios como sea necesario.

Esta función sólo funciona con un datastore remoto (cliente/servidor o conexión [Open datastore](#)).

## Ejemplo

```
// orden con fórmula
$sortedEntitySelection:=$entitySelection.orderBy("firstName asc, salary desc")
$sortedEntitySelection:=$entitySelection.orderBy("firstName")

// orden con collection con o sin órdenes de clasificación
$orderColl:=New collection
$orderColl.push(New object("propertyPath";"firstName";"descending";False))
$orderColl.push(New object("propertyPath";"salary";"descending";True))
$sortedEntitySelection:=$entitySelection.orderBy($orderColl)

$orderColl:=New collection
$orderColl.push(New object("propertyPath";"manager.lastName"))
$orderColl.push(New object("propertyPath";"salary"))
$sortedEntitySelection:=$entitySelection.orderBy($orderColl)
```

## .orderByFormula()

### ► Histórico

.orderByFormula( *formulaString* : Text { ; *sortOrder* : Integer } { ; *settings* : Object} ) : 4D.EntitySelection  
.orderByFormula( *formulaObj* : Object { ; *sortOrder* : Integer } { ; *settings* : Object} ) : 4D.EntitySelection

Parámetros	Tipo		Descripción
<i>formulaString</i>	Texto	->	Cadena formula
<i>formulaObj</i>	Objeto	->	Objeto formula
<i>sortOrder</i>	Integer	->	dk ascending (por defecto) o dk descending
parámetros	Objeto	->	Parámetros de la fórmula
Resultado	4D.EntitySelection	<-	Nueva entity selection ordenada

### Descripción

La función `.orderByFormula()` devuelve una nueva entity selection ordenada que contiene todas las entidades de la entity selection en el orden definido a través de los parámetros *formulaString* o *formulaObj* y, opcionalmente, *sortOrder* y *settings*.

Las entidades de una colección de entidades a las que se accede por medio de [ ] no se recargan desde la base de datos.

Puede utilizar un parámetro *formulaString* o un parámetro *formulaObj*:

- *formulaString*: se pasa una expresión 4D como "Year of(this.birthDate)".
- *formulaObj*: pase un objeto fórmula válido creado con el comando `Formula` o `Formula from string`.

La fórmula de *formulaString* o *formulaObj* se ejecuta para cada entidad de la entity selection y su resultado se utiliza para definir la posición de la entidad en la entity selection devuelta. El resultado debe ser de un tipo ordenable (booleano, fecha, número, texto, hora, null).

Un resultado null es siempre el valor más pequeño.

Por defecto, si se omite el parámetro *sortOrder*, la selección de entidades resultante se clasifica en orden ascendente. Opcionalmente, puede pasar uno de los siguientes valores en el parámetro *sortOrder*:

Constante	Valor	Comentario
dk ascending	0	Orden ascendente (por defecto)
dk descending	1	Orden descendente

Dentro de `formulaString` o `formulaObj`, la entidad procesada y, por tanto, sus atributos están disponibles a través del comando `This` (por ejemplo, `This.lastName`).

Puede pasar parámetros a la fórmula utilizando la propiedad `args/0>` (objeto) del parámetro `<code>settings`: la fórmula recibe el objeto `settings.args` en \$1.

## Ejemplo 1

Clasificar a los alumnos mediante una fórmula dada como texto:

```
var $es1; $es2 : cs.StudentsSelection
$es1:=ds.Students.query("nationality=:1;"French")
$es2:=$es1.orderByFormula("length(this.lastname)") //ascendente por defecto
$es2:=$es1.orderByFormula("length(this.lastname);dk descending")
```

El mismo orden de clasificación pero utilizando un objeto fórmula:

```
var $es1; $es2 : cs.StudentsSelection
var $formula : Object
$es1:=ds.Students.query("nationality=:1;"French")
$formula:=Formula(Length(This.lastname))
$es2:=$es1.orderByFormula($formula) // ascendente por defecto
$es2:=$es1.orderByFormula($formula;dk descending)
```

## Ejemplo 2

Una fórmula se da como un objeto fórmula con parámetros; el objeto `settings.args` se recibe como \$1 en el método `computeAverage`.

En este ejemplo, el campo objeto "marks" de la dataClass `Students` contiene las notas de los estudiantes para cada asignatura. Se utiliza un solo objeto fórmula para calcular la nota media de un alumno con diferentes coeficientes para `schoolA` y `schoolB`.

```
var $es1; $es2 : cs.StudentsSelection
var $formula; $schoolA; $schoolB : Object
$es1:=ds.Students.query("nationality=:1;"French")
$formula:=Formula(computeAverage($1))

$schoolA:=New object() //settings object
$schoolA.args:=New object("english";1;"math";1;"history";1) // Coeficientes para calcular una media

//Ordenar a los estudiantes según los criterios de la escuela A
$es2:=$es1.entitySelection.orderByFormula($formula;$schoolA)

$schoolB:=New object() //object settings
$schoolB.args:=New object("english";1;"math";2;"history";3) // Coeficientes para calcular un promedio

//Ordenar a los estudiantes según los criterios de la escuela B
$es2:=$es1.entitySelection.orderByFormula($formula;dk descending;$schoolB)
```

```

// 
// método computeAverage
// -----
#DECLARE ($coefList : Object) -> $result : Integer
var $subject : Text
var $average; $sum : Integer

$average:=0
$sum:=0

For each($subject;$coefList)
    $sum:=$sum+$coefList[$subject]
End for each

For each($subject;This.marks)
    $average:=$average+(This.marks[$subject]*$coefList[$subject])
End for each

$result:=$average/$sum

```

## .query()

► Histórico

.query( *queryString* : Text { ; ...*value* : any } { ; *querySettings* : Object } ) : 4D.EntitySelection  
 .query( *formula* : Object { ; *querySettings* : Object } ) : 4D.EntitySelection

Parámetros	Tipo		Descripción
<i>queryString</i>	Texto	->	Criterios de búsqueda como cadena
<i>formula</i>	Objeto	->	Criterios de búsqueda como objeto fórmula
<i>value</i>	any	->	Valor(es) a utilizar para los marcadores de posición indexados
<i>querySettings</i>	Objeto	->	Opciones de búsqueda: parameters, attributes, args, allowFormulas, context, queryPath, queryPlan
Resultado	4D.EntitySelection	<-	Nueva selección de entidades formada por las entidades de la entity selection que cumplen los criterios de búsqueda especificados en <i>queryString</i> o <i>formula</i>

### Descripción

La función `.query()` busca entidades que cumplan los criterios de búsqueda especificados en *queryString* o *formula* y (opcionalmente) *value*(s) entre todas las entidades de la selección de entidades, y devuelve un nuevo objeto de tipo `EntitySelection` que contiene todas las entidades encontradas. Se aplica carga diferida.

Las entidades de una colección de entidades a las que se accede por medio de [ ] no se recargan desde la base de datos.

Si no se encuentran entidades coincidentes, se devuelve una `EntitySelection` vacía.

Si no se encuentran entidades coincidentes, se devuelve una `EntitySelection` vacía.

Por defecto, si se omite la declaración `order by` en *queryString*, la entity selection devuelta es [no ordenada](#). Sin embargo, tenga en cuenta que, en modo Cliente/Servidor, se comporta como una selección de entidades ordenada (las entidades se añaden al final de la selección).

### Ejemplo 1

```
var $entitySelectionTemp : cs.EmployeeSelection  
$entitySelectionTemp:=ds.Employee.query("lastName = :1";"M@")  
Form.emps:=$entitySelectionTemp.query("manager.lastName = :1";"S@")
```

## Ejemplo 2

Para obtener información detallada sobre cómo construir una consulta utilizando los parámetros `queryString`, `value<querySettings`, consulte la descripción de la función de `dataClass .query()`.

### Ver también

Se pueden encontrar más ejemplos de búsquedas en la página [.query\(\) DataClass](#).

## .queryPath

► Histórico

`.queryPath` : Text

### Descripción

La propiedad `.queryPath` contiene una descripción detallada de la búsqueda tal y como fue realizada por 4D. Esta propiedad está disponible para los objetos de tipo `EntitySelection` generados a través de búsquedas si la propiedad `"queryPath":true` fue pasada en el parámetro `querySettings` de la función [.query\(\)](#).

Para más información, consulte el párrafo `querySettings` en la página [.query\(\) Dataclass](#).

## .queryPlan

► Histórico

`.queryPlan` : Text

### Descripción

La propiedad `.queryPlan` contiene una descripción detallada de la búsqueda justo antes de su ejecución (es decir, la búsqueda planeada). Esta propiedad está disponible para los objetos de tipo `EntitySelection` generados a través de búsquedas si la propiedad `"queryPlan":true` fue pasada en el parámetro `querySettings` de la función [.query\(\)](#).

Para más información, consulte el párrafo `querySettings` en la página [.query\(\) Dataclass](#).

## .refresh()

► Histórico

`.refresh()`

Parámetros	Tipo	Descripción	-----	----	::	-----		No requiere ningún parámetro
------------	------	-------------	-------	------	----	-------	--	------------------------------

### Descripción

Esta función sólo funciona con un datastore remoto (cliente/servidor o conexión [Open datastore](#)).

La función `.refresh()` invalida inmediatamente los datos de la entity selection en la caché local de ORDA para que la próxima vez que 4D requiera la entity selection, ésta sea recargada desde la base.

Por defecto, la caché local de ORDA se invalida después de 30 segundos. En el contexto de las aplicaciones cliente/servidor que utilizan tanto ORDA como el lenguaje clásico, este método le permite asegurarse de que una aplicación remota siempre funcionará con los datos más recientes.

## Ejemplo 1

En este ejemplo, el código clásico y el código ORDA modifican los mismos datos simultáneamente:

```
//En un 4D remoto

var $selection : cs.StudentsSelection
var $student : cs.StudentsEntity

$selection:=ds.Students.query("lastname=:1";"Collins")
//La primera entidad se carga en la caché de ORDA
$student:=$selection.first()

//Actualización con un 4D clásico, la caché ORDA no es consciente de si
QUERY([Students];[Students]lastname="Collins")
[Students]lastname:="Colin"
SAVE RECORD([Students])

//para obtener la última versión, hay que invalidar la caché de ORDA
$selection.refresh()
// Aunque la caché no haya caducado, la primera entidad se recarga desde el disco
$student:=$selection.first()

// $student.lastname contains "Colin"
```

## Ejemplo 2

En este ejemplo, el código clásico y el código ORDA modifican los mismos datos simultáneamente:

```
// Método formulario: Case of
:(Form event code=On Load)
  Form.students:=ds.Students.all()
End case
//
//
// En client #1, el usuario carga, actualiza y guarda la primera entidad
// En client #2, el usuario carga, actualiza y guarda la misma entidad
//
//
// En client #1:
Form.students.refresh() // Invalida la caché ORDA para la entity selection Form.students
// El contenido del list box se refresca desde la base con la actualización realizada por el cliente #
```

## .selected()

► Histórico

.selected( *selectedEntities* : 4D.EntitySelection ) : Object

Parámetros	Tipo		Descripción
selectedEntities	4D.EntitySelection	->	Selección de entidades con entidades para las cuales conocer el rango en la selección de entidades
Resultado	Objeto	<-	Rango(s) de entidades seleccionadas en la selección de entidades

### Descripción

La función `.selected()` devuelve un objeto que describe la(s) posición(es) de *selectedEntities* en la selección de entidades original.

Las entidades de una colección de entidades a las que se accede por medio de [ ] no se recargan desde la base de datos.

Pase en el parámetro `selectedEntities` una selección de entidades que contenga entidades de las que deseé conocer la posición en la selección de entidades original. `selectedEntities` debe ser una selección de entidades que pertenezca a la misma clase de datos que la selección de entidades original, de lo contrario se produce un error 1587 - "La selección de entidades procede de una clase de datos incompatible".

## Resultado

El objeto devuelto contiene las siguientes propiedades:

Propiedad	Tipo	Descripción
<code>ranges</code>	Collection	Colección de objetos de rango
<code>ranges[]</code> .start	Integer	Primer índice de entidad en el rango
<code>ranges[]</code> .end	Integer	Último índice de la entidad en el rango

Si una propiedad `ranges` contiene una sola entidad, `start = end`. El índice comienza en 0.

La función devuelve una colección vacía en la propiedad `ranges` si la selección de entidades original o la selección de entidades `selectedEntities` está vacía.

## Ejemplo

```
var $invoices; $cashSel; $creditSel : cs.Invoices
var $result1; $result2 : Object

$invoices:=ds.Invoices.all()

$cashSelection:=ds.Invoices.query("payment = :1"; "Cash")
$creditSel:=ds.Invoices.query("payment IN :1"; New collection("Cash"; "Credit Card"))

$result1:=$invoices.selected($cashSelection)
$result2:=$invoices.selected($creditSel)

//$result1 = {ranges: [{start:0;end:0},{start:3;end:3},{start:6;end:6}]}
//$result2 = {ranges: [{start:0;end:1},{start:3;end:4},{start:6;end:7}]}
```

## .slice()

► Histórico

`.slice( startFrom : Integer { ; end : Integer } ) : 4D.EntitySelection`

Parámetros	Tipo		Descripción
<code>startFrom</code>	Integer	->	Índice para iniciar la operación (incluído)
<code>end</code>	Integer	->	Índice final (no incluido)
Resultado	4D.EntitySelection	<-	Nueva entity selection que contiene la entidades extraídas (shallow copy)

## Descripción

La función `.slice()` devuelve una parte de una selección de entidades en una nueva entity selection, seleccionada desde el índice `startFrom` hasta el índice `end` (`end` no se incluye) o hasta la última entidad de la entity selection. Este método devuelve una shallow copy (copia superficial) de la entity selection (utiliza las mismas referencias de entidades).

Las entidades de una colección de entidades a las que se accede por medio de [ ] no se recargan desde la base de datos.

La entity selection devuelta contiene las entidades especificadas por *startFrom* y todas las entidades subsiguientes hasta, pero sin incluir, la entidad especificada por *end*. Si sólo se especifica el parámetro *startFrom*, la entity selection devuelta contiene todas las entidades entre *startFrom* y la última entidad de la entity selection original.

- Si *startFrom* < 0, se recalcula como *startFrom*:=*startFrom*+*length*(se considera el desplazamiento desde el final de la entity selection). Si el valor calculado < 0, *startFrom* toma el valor 0.
- Si *startFrom* >= *length*, la función devuelve una selección de entidades vacía.
- Si *end* < 0, se recalcula como *end*:=*end*+*length*.
- Si *end* < *startFrom* (valores pasados o calculados), el método no hace nada.

`.sum()` devuelve 0 si la entity selection está vacía.

## Ejemplo 1

Si la entity selection contiene entidades que se han eliminado mientras tanto, también se devuelven.

```
var $sel; $sliced : cs.EmployeeSelection
$sel:=ds.Employee.query("salary > :1";50000)
$sliced:=$sel.slice(0;9) //
```

## Ejemplo 2

Quiere obtener una selección de las primeras 9 entidades de la entity selection:

```
var $slice : cs.EmployeeSelection
$slice:=ds.Employee.all().slice(-1;-2) //intenta devolver entidades del índice 9 al 8, pero como 9 > 8,
```

## `.sum( )`

► Histórico

`.sum( attributePath : Text ) : Real`

Parámetros	Tipo		Descripción
attributePath	Texto	->	Ruta del atributo que se utilizará para el cálculo
Resultado	Real	<-	Suma de los valores de la entity selection

### Descripción

La función `.toCollection()` crea y devuelve una colección donde cada elemento es un objeto que contiene un conjunto de propiedades y valores correspondientes a los nombres y valores de los atributos de la entity selection.

La función `.sum()` devuelve la suma de todos los valores de *attributePath* en la entity selection.

La suma sólo puede hacerse en valores de tipo numérico. Si *attributePath* es una propiedad objeto, sólo se tienen en cuenta los valores numéricos para el cálculo (se ignoran otros tipos de valores). En este caso, si *attributePath* lleva a una propiedad que no existe en el objeto o no contiene ningún valor numérico, `.sum()` devuelve 0.

Se devuelve un error si:

- *attributePath* no es un atributo numérico o un objeto,
- *>attributePath* es un atributo relativo,
- *attributePath* no se encuentra en la clase de datos de la entity selection.

## Ejemplo

```
var $sel : cs.EmployeeSelection
var $sum : Real

$sel:=ds.Employee.query("salary < :1";20000)
$sum:=$sel.sum("salary")
```

## .toCollection( )

► Histórico

.toCollection( { options : Integer { ; begin : Integer { ; howMany : Integer } } } ) : Collection  
.toCollection( filterString : Text {; options : Integer { ; begin : Integer { ; howMany : Integer } } } ) : Collection  
.toCollection( filterCol : Collection {; options : Integer { ; begin : Integer { ; howMany : Integer } } } ) : Collection

Parámetros	Tipo		Descripción
filterString	Texto	->	Cadena con la(s) ruta(s) de atributos de la entidad a extraer
filterCol	Collection	->	Colección de rutas de atributos de entidad a extraer
options	Integer	->	dk with primary key : añade la llave primaria dk with stamp< : añade el sello
begin	Integer	->	Designa el índice inicial
howMany	Integer	->	Número de entidades a extraer
Resultado	Collection	<-	Colección de objetos que contienen atributos y valores de la selección de entidades

### Descripción

La función `.toCollection()` crea y devuelve una colección donde cada elemento es un objeto que contiene un conjunto de propiedades y valores correspondientes a los nombres y valores de los atributos de la entity selection.

Si no se pasa ningún parámetro de filtro o si el primer parámetro contiene una cadena vacía o "\*", se extraen todos los atributos. Los atributos con la propiedad `kind` como "relatedEntity" se extraen con el formulario simple: un objeto con la propiedad `__KEY` (llave primaria). Los atributos con la propiedad `kind` como "relatedEntities" no se extraen.

O bien, puede designar los atributos de la entidad a extraer utilizando un parámetro de filtro. Puede utilizar uno de estos dos filtros:

- `filterString<` --una cadena con rutas de propiedades separadas por comas: "propertyPath1, propertyPath2, ...".
- `filterCol:` --una colección de cadenas que contiene la rutas de propiedades: ["propertyPath1","propertyPath2",...]

Si se especifica un filtro para un atributo de tipo `relatedEntity`:

- `propertyPath = "relatedEntity"` -> se extrae con una forma simple
- `propertyPath = "relatedEntity.*"` -> se extraen todas las propiedades
- `propertyPath = "relatedEntity.propertyName1, relatedEntity.propertyName2, ..."` -> sólo se extraen esas propiedades

Si se especifica un filtro para un atributo de tipo `relatedEntity`:

- `propertyPath = "relatedEntities.*"` -> se extraen todas las propiedades
- `propertyPath = "relatedEntities.propertyName1, relatedEntities.propertyName2, ..."` -> sólo se extraen esas propiedades

Si se especifica un filtro para un atributo de tipo `relatedEntities`:

El parámetro `begin` permite indicar el índice de inicio de las entidades a extraer. Puede pasar cualquier valor entre 0 y la longitud de la entity selection -1.

El parámetro `howMany` permite especificar el número de entidades a extraer, empezando por la especificada en `begin`. Las entidades descartadas no se devuelven pero se tienen en cuenta en `howMany`. Por ejemplo, si `howMany= 3` y hay 1 entidad descartada, sólo se extraen 2 entidades.

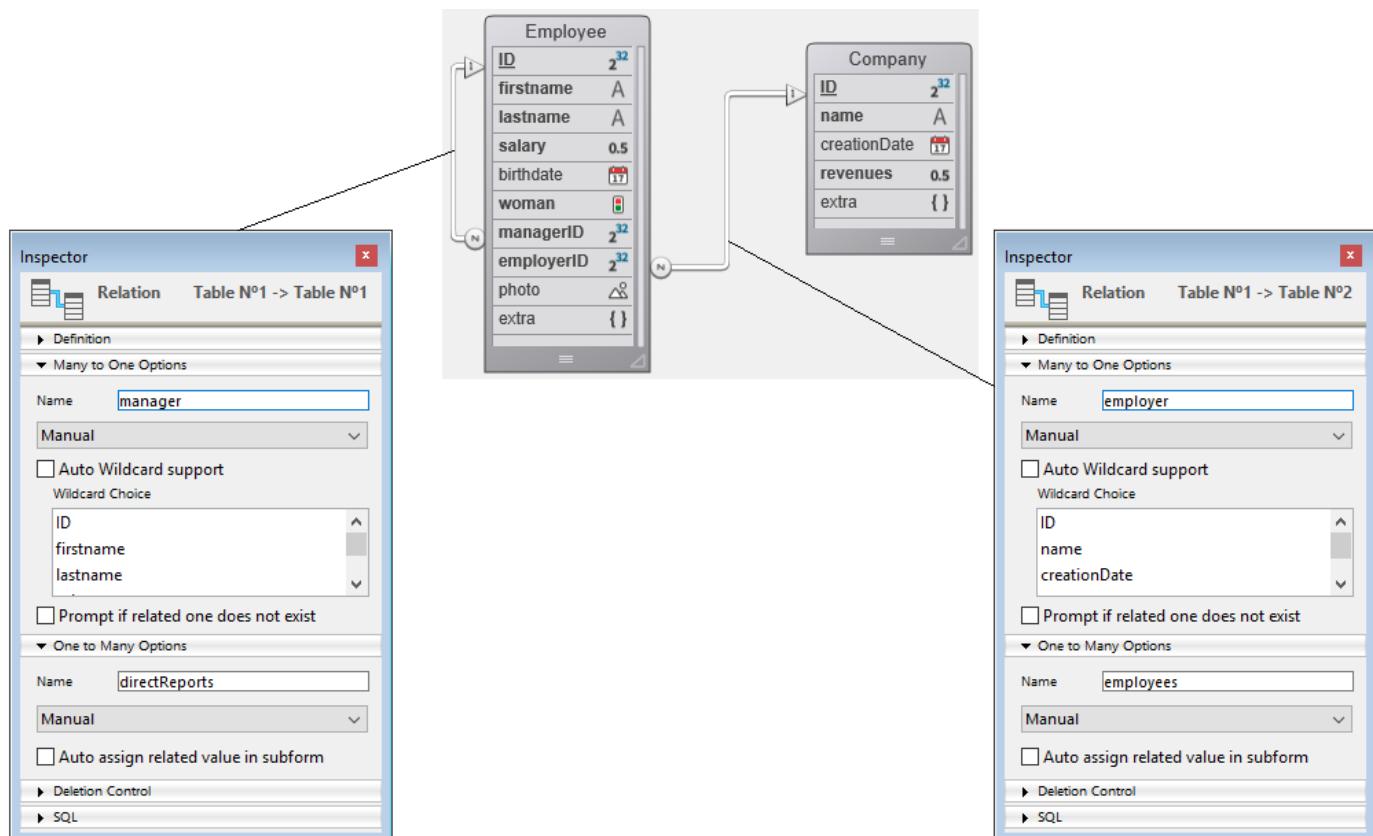
Si el "kind" de `attributeName` es `storage : .attributeName` devuelve una colección de valores del mismo tipo que `attributeName`.

Si `howMany > longitud de la entity selection`, el método devuelve (`length - begin`) objetos.

- la entity selection está vacía, o
- `begin` es mayor que la longitud de la entity selection.

## Ejemplo 1

En todos los ejemplos de esta sección se utilizará la siguiente estructura:



Ejemplo sin parámetros de filtro ni de opciones:

```

var $employeesCollection : Collection
var $employees : cs.EmployeeSelection

$employeesCollection:=New collection
$employees:=ds.Employee.all()
$employeesCollection:=$employees.toCollection()

```

Devuelve:

```
[
  {
    "ID": 416,
    "firstName": "Gregg",
    "lastName": "Wahl",
    "salary": 79100,
    "birthDate": "1963-02-01T00:00:00.000Z",
    "woman": false,
    "managerID": 412,
    "employerID": 20,
    "photo": "[object Picture]",
    "extra": null,
    "employer": {
      "__KEY": 20
    },
    "manager": {
      "__KEY": 412
    }
  },
  {
    "ID": 417,
    "firstName": "Irma",
    "lastName": "Durham",
    "salary": 47000,
    "birthDate": "1992-06-16T00:00:00.000Z",
    "woman": true,
    "managerID": 412,
    "employerID": 20,
    "photo": "[object Picture]",
    "extra": null,
    "employer": {
      "__KEY": 20
    },
    "manager": {
      "__KEY": 412
    }
  }
]
]
```

## Ejemplo 2

Devuelve:

```
var $employeesCollection : Collection
var $employees : cs.EmployeeSelection

$employeesCollection:=New collection
$employees:=ds.Employee.all()
$employeesCollection:=$employees.toCollection("");dk with primary key+dk with stamp)
```

Devuelve:

```
[
  {
    "__KEY": 416,
    "__STAMP": 1,
    "ID": 416,
    "firstName": "Gregg",
    "lastName": "Wahl",
    "salary": 79100,
    "birthDate": "1963-02-01T00:00:00.000Z",
    "woman": false,
    "managerID": 412,
    "employerID": 20,
    "photo": "[object Picture]",
    "extra": null,
    "employer": {
      "__KEY": 20
    },
    "manager": {
      "__KEY": 412
    }
  },
  {
    "__KEY": 417,
    "__STAMP": 1,
    "ID": 417,
    "firstName": "Irma",
    "lastName": "Durham",
    "salary": 47000,
    "birthDate": "1992-06-16T00:00:00.000Z",
    "woman": true,
    "managerID": 412,
    "employerID": 20,
    "photo": "[object Picture]",
    "extra": null,
    "employer": {
      "__KEY": 20
    },
    "manager": {
      "__KEY": 412
    }
  }
]
```

### Ejemplo 3

Devuelve:

```
var $employeesCollection; $filter : Collection
var $employees : cs.EmployeeSelection

$employeesCollection:=New collection
$filter:=New collection

$filter.push("firstName")
$filter.push("lastName")

$employees:=ds.Employee.all()
$employeesCollection:=$employees.toCollection($filter;0;0;2)
```

Devuelve:

```
[  
  {  
    "firstName": "Gregg",  
    "lastName": "Wahl"  
  },  
  {  
    "firstName": "Irma",  
    "lastName": "Durham"  
  }  
]
```

#### Ejemplo 4

Devuelve:

```
var $employeesCollection : Collection  
$employeesCollection:=New collection  
$employeesCollection:=$employees.toCollection("firstName,lastName,employer")
```

Ejemplo con el tipo `relatedEntity` con una forma simple:

```
[  
  {  
    "firstName": "Gregg",  
    "lastName": "Wahl",  
    "employer": {  
      "__KEY": 20  
    }  
  },  
  {  
    "firstName": "Irma",  
    "lastName": "Durham",  
    "employer": {  
      "__KEY": 20  
    }  
  },  
  {  
    "firstName": "Lorena",  
    "lastName": "Boothe",  
    "employer": {  
      "__KEY": 20  
    }  
  }  
]
```

#### Ejemplo 5

devuelve:

```
var $employeesCollection; $coll : Collection  
$employeesCollection:=New collection  
$coll:=New collection("firstName";"lastName")  
$employeesCollection:=$employees.toCollection($coll)
```

Devuelve:

```
[  
  {  
    "firstName": "Joanna",  
    "lastName": "Cabrera"  
  },  
  {  
    "firstName": "Alexandra",  
    "lastName": "Coleman"  
  }  
]
```

## Ejemplo 6

Devuelve:

```
var $employeesCollection; $coll : Collection  
$employeesCollection:=New collection  
$coll:=New collection  
$coll.push("firstName")  
$coll.push("lastName")  
$coll.push("employer.*")  
$employeesCollection:=$employees.toCollection($coll)
```

Devuelve:

```
[  
  {  
    "firstName": "Gregg",  
    "lastName": "Wahl",  
    "employer": {  
      "ID": 20,  
      "name": "India Astral Secretary",  
      "creationDate": "1984-08-25T00:00:00.000Z",  
      "revenues": 12000000,  
      "extra": null  
    }  
  },  
  {  
    "firstName": "Irma",  
    "lastName": "Durham",  
    "employer": {  
      "ID": 20,  
      "name": "India Astral Secretary",  
      "creationDate": "1984-08-25T00:00:00.000Z",  
      "revenues": 12000000,  
      "extra": null  
    }  
  },  
  {  
    "firstName": "Lorena",  
    "lastName": "Boothe",  
    "employer": {  
      "ID": 20,  
      "name": "India Astral Secretary",  
      "creationDate": "1984-08-25T00:00:00.000Z",  
      "revenues": 12000000,  
      "extra": null  
    }  
  }  
]
```

## Ejemplo 7

Devuelve:

```
var $employeesCollection : Collection
$employeesCollection:=New collection
$employeesCollection:=$employees.toCollection("firstName, lastName, employer.name")
```

```
[  
  {  
    "firstName": "Gregg",  
    "lastName": "Wahl",  
  
    "employer": {  
      "name": "India Astral Secretary"  
    }  
  },  
  {  
    "firstName": "Irma",  
    "lastName": "Durham",  
    "employer": {  
      "name": "India Astral Secretary"  
    }  
  },  
  {  
    "firstName": "Lorena",  
    "lastName": "Boothe",  
    "employer": {  
      "name": "India Astral Secretary"  
    }  
  }]  
}]
```

## Ejemplo 8

Ejemplo con la extracción de algunas las propiedades de relatedEntity:

```
var $employeesCollection : Collection
$employeesCollection:=New collection
$employeesCollection:=$employees.toCollection("firstName, lastName, directReports.firstName")
```

Devuelve:

```
[
  {
    "firstName": "Gregg",
    "lastName": "Wahl",
    "directReports": []
  },
  {
    "firstName": "Mike",
    "lastName": "Phan",
    "directReports": [
      {
        "firstName": "Gary"
      },
      {
        "firstName": "Sadie"
      },
      {
        "firstName": "Christie"
      }
    ]
  },
  {
    "firstName": "Gary",
    "lastName": "Reichert",
    "directReports": [
      {
        "firstName": "Rex"
      },
      {
        "firstName": "Jenny"
      },
      {
        "firstName": "Lowell"
      }
    ]
  }
]
```

## Ejemplo 9

Devuelve:

```
var $employeesCollection : Collection
$employeesCollection:=New collection
$employeesCollection:=$employees.toCollection("firstName, lastName, directReports.*")
```

```
[
  {
    "firstName": "Gregg",
    "lastName": "Wahl",
    "directReports": []
  },
  {
    "firstName": "Mike",
    "lastName": "Phan",
    "directReports": [
      {
        "ID": 425,
        "firstName": "Gary",
        "lastName": "Reichert",
        "directReports": []
      }
    ]
  }
]
```

```
        "salary": 65800,
        "birthDate": "1957-12-23T00:00:00.000Z",
        "woman": false,
        "managerID": 424,
        "employerID": 21,
        "photo": "[object Picture]",
        "extra": null,
        "employer": {
            "__KEY": 21
        },
        "manager": {
            "__KEY": 424
        }
    },
    {
        "ID": 426,
        "firstName": "Sadie",
        "lastName": "Gallant",
        "salary": 35200,
        "birthDate": "2022-01-03T00:00:00.000Z",
        "woman": true,
        "managerID": 424,
        "employerID": 21,
        "photo": "[object Picture]",
        "extra": null,
        "employer": {
            "__KEY": 21
        },
        "manager": {
            "__KEY": 424
        }
    }
],
},
{
    "firstName": "Gary",
    "lastName": "Reichert",
    "directReports": [
        {
            "ID": 428,
            "firstName": "Rex",
            "lastName": "Chance",
            "salary": 71600,
            "birthDate": "1968-08-09T00:00:00.000Z",
            "woman": false,

            "managerID": 425,
            "employerID": 21,
            "photo": "[object Picture]",
            "extra": null,
            "employer": {
                "__KEY": 21
            },
            "manager": {
                "__KEY": 425
            }
        },
        {
            "ID": 429,
            "firstName": "Jenny",
            "lastName": "Parks",
            "salary": 51300,
            "birthDate": "1984-05-25T00:00:00.000Z",
            "woman": true,
            "managerID": 425,
            "employer": {
                "__KEY": 21
            },
            "photo": "[object Picture]"
        }
    ]
}
]
```

```
"employerID": 21,
"photo": "[object Picture]",
"extra": null,
"employer": {
    "__KEY": 21
},
"manager": {
    "__KEY": 425
}
}
]
}
```

# File

Los objetos `File` se crean con el comando `File`. Contienen referencias a archivos de disco que pueden o no existir realmente en el disco. Por ejemplo, cuando se ejecuta el comando `File` para crear un nuevo archivo, se crea un objeto `File` válido, pero en realidad no se almacena nada en el disco hasta que se llama a la función `file.create()`.

## Ejemplo

El siguiente ejemplo crea un archivo de preferencias en la carpeta del proyecto:

```
var $created : Boolean  
$created:=File("/PACKAGE/SpecialPrefs/"+Current user+".myPrefs").create()
```

## Objeto File

<code>.copyTo( destinationFolder : 4D.Folder { ; newName : Text } { ; overwrite : Integer } ) : 4D.File</code> copia el objeto <code>File</code> en el <code>destinationFolder</code> especificado>
<code>.create() : Boolean</code> creates a file on disk according to the properties of the <code>File</code> object
<code>.createAlias( destinationFolder : 4D.Folder ; aliasName : Text { ; aliasType : Integer } ) : 4D.File</code> creates an alias (macOS) or a shortcut (Windows)
<code>.creationDate : Date</code> la fecha de creación del archivo
<code>.creationTime : Time</code> la hora de creación del archivo
<code>.delete( )</code> deletes the file
<code>.exists : Boolean</code> true si el archivo existe en el disco
<code>.extension : Text</code> la extensión del nombre del archivo (si lo hay)
<code>.fullName : Text</code> el nombre completo del archivo, incluyendo su extensión (si la hay)
<code>.getAppInfo() : Object</code> returns the contents of a .exe, .dll or .plist file information as an object
<code>.getContent( ) : 4D.Blob</code> devuelve un objeto <code>4D.Blob</code> que contiene todo el contenido de un archivo
<code>.getIcon( { size : Integer } ) : Picture</code> el ícono del archivo
<code>.getText( { charSetName : Text { ; breakMode : Integer } } ) : Text</code> <code>getText( \$charSetName : Integer \$breakMode : Integer ) : Text</code>

`.getRaw( charSetName : Text , breakMode : Integer ) : Text`

devuelve el contenido del archivo como texto

**.hidden : Boolean**

true si el archivo está configurado como "oculto" a nivel de sistema

**.isAlias : Boolean**

true si el archivo es un alias, un acceso directo o un enlace simbólico

**.isFile : Boolean**

siempre verdadero para un archivo

**.isFolder : Boolean**

siempre falso para un archivo

**.isWritable : Boolean**

true si el archivo existe en el disco y es escribible

**.modificationDate : Date**

la fecha de la última modificación del archivo

**.modificationTime : Time**

la hora de la última modificación del archivo

**.moveTo( destinationFolder : 4D.Folder { ; newName : Text } ) : 4D.File**

moves or renames the `File` object into the specified `destinationFolder`

**.name : Text**

el nombre del archivo, sin extensión (si la hay)

**.original : 4D.File**

**.original : 4D.Folder**

**.parent : 4D.Folder**

el objeto carpeta padre del archivo

**.path : Text**

la ruta POSIX del archivo

**.platformPath : Text**

la ruta del archivo expresada con la sintaxis de la plataforma actual

**.rename( newName : Text ) : 4D.File**

renames the file with the name you passed in `newName` and returns the renamed `File` object

**.setAppInfo( info : Object )**

writes the `info` properties as information contents of a .exe, .dll or .plist file

**.setContent ( content : Blob )**

rewrites the entire content of the file using the data stored in the `content` BLOB

**.setText ( text : Text {; charSetName : Text { ; breakMode : Integer } } )**

**.setText ( text : Text {; charSetNum : Integer { ; breakMode : Integer } } )**

writes `text` as the new contents of the file

**.size : Real**

el tamaño del archivo expresado en bytes

# File

► Histórico

`File ( path : Text { ; pathType : Integer }{ ; * } ) : 4D.File`

`File ( fileConstant : Integer { ; * } ) : 4D.File`

Parámetros	Tipo		Descripción
path	Texto	->	Ruta del archivo
fileConstant	Integer	->	Constante del archivo 4D
pathType	Integer	->	<code>fk posix path</code> (por defecto) o <code>fk platform path</code>
*		->	* para devolver el archivo de la base local
Resultado	4D.File	<-	Nuevo objeto de archivo

## Descripción

El comando `File` crea y devuelve un nuevo objeto de tipo `4D.File`. El comando acepta dos sintaxis:

`File ( path { ; pathType } { ; * } )`

En el parámetro `path`, pase una ruta de archivo. Puede utilizar una cadena personalizada o un filesystem (por ejemplo, `"/DATA/myfile.txt"`).

Sólo se admiten nombres de ruta absolutos con el comando `File`.

Por defecto, 4D espera una ruta expresada con la sintaxis POSIX. Si trabaja con los nombres de ruta de plataforma (Windows o macOS), debe declararlo utilizando el parámetro `pathType`. Las siguientes constantes están disponibles:

Constante	Valor	Comentario
<code>fk platform path</code>	1	Ruta expresada con una sintaxis específica de la plataforma (obligatoria en caso de nombre de ruta de plataforma)
<code>fk posix path</code>	0	Ruta expresada con sintaxis POSIX (por defecto)

`File ( fileConstant { ; * } )`

En el parámetro `fileConstant`, pase un archivo 4D interno o sistema, utilizando una de las siguientes constantes:

Constante	Valor	Comentario
<code>Backup history file</code>	19	Archivo de historial de copias de seguridad (ver Archivos de configuración y rastreo). Se almacena en la carpeta de destino de la copia de seguridad.
<code>Backup log file</code>	13	Archivo historial de copias de seguridad actual. Almacenado en la carpeta Logs de la aplicación.
<code>Backup settings file</code>	1	Archivo backup.4DSettings por defecto (formato xml), almacenado en la carpeta Settings del proyecto
<code>Backup settings file for data</code>	17	archivo backup.4DSettings del archivo de datos (formato xml), almacenado en la carpeta Settings de la carpeta data
<code>Build application log file</code>	14	Archivo de historial actual en formato xml del generador de aplicaciones. Almacenado en la carpeta Logs.
<code>Build application</code>	20	Default settings file of the application builder ("buildApp.4DSettings"). Almacenado en la carpeta Settings del proyecto

application Constante	Valor	Carpeta Settings del proyecto. Comentario
Compacting log file	6	Log file of the most recent compacting done with the Compact data file command or the Maintenance and security center. Almacenado en la carpeta Logs.
Current backup settings file	18	backup.4DSettings file currently used by the application. It can be the backup settings file (default) or a custom user backup settings file defined for the data file
Debug log file	12	Log file created by the <code>SET DATABASE PARAMETER(Debug log recording)</code> command. Almacenado en la carpeta Logs.
Diagnostic log file	11	Log file created by the <code>SET DATABASE PARAMETER(Diagnostic log recording)</code> command. Almacenado en la carpeta Logs.
Directory file	16	directory.json file, containing the description of users and groups (if any) for the project application. It can be located either in the user settings folder (default, global to the project), or in the data settings folder (specific to a data file).
HTTP debug log file	9	Log file created by the <code>WEB SET OPTION(Web debug log)</code> command. Almacenado en la carpeta Logs.
HTTP log file	8	Log file created by the <code>WEB SET OPTION(Web log recording)</code> command. Almacenado en la carpeta Logs.
IMAP Log file	23	Log file created by the <code>SET DATABASE PARAMETER(IMAP Log)</code> command. Almacenado en la carpeta Logs.
Last backup file	2	Último archivo de copia de seguridad, llamado <applicationName>[bkpNum].4BK, almacenado en una ubicación personalizada.
Last journal integration log file	22	Full pathname of the last journal integration log file (stored in the Logs folder of the restored application), if any. This file is created, in auto-repair mode, as soon as a log file integration occurred
Repair log file	7	Log file of database repairs made on the database in the Maintenance and Security Center (MSC). Almacenado en la carpeta Logs.
Request log file	10	Standard client/server request log file (excluding Web requests) created by the <code>SET DATABASE PARAMETER(4D Server log recording)</code> or <code>SET DATABASE PARAMETER(Client log recording)</code> commands. If executed on the server, the server log file is returned (stored in the Logs folder on the server). If executed on the client, the client log file is returned (stored in the client local Logs folder).
SMTP log file	15	Log file created by the <code>SET DATABASE PARAMETER(SMTP Log)</code> command. Almacenado en la carpeta Logs.
User settings file	3	settings.4DSettings file for all data files, stored in Preferences folder next to structure file if enabled.
User settings file for data	4	settings.4DSettings file for current data file, stored in Preferences folder next to the data file.
Verification log file	5	Log files created by the <code>VERIFY CURRENT DATA FILE</code> and <code>VERIFY DATA FILE</code> commands or the Maintenance and Security Center (MSC). Almacenado en la carpeta Logs.

If the target `fileConstant` does not exist, a null object is returned. No se produce ningún error.

If the command is called from a component, pass the optional `*` parameter to get the path of the host database. Otherwise, if you omit the `*` parameter, a null object is always returned.

## 4D.File.new()

► Histórico

4D.File.new ( `path : Text { ; pathType : Integer }{ ; * }` ) : 4D.File

`4D.File.new ( fileConstant : Integer { ; * } ) : 4D.File`

## Descripción

La función `4D.File.new()` crea y devuelve un nuevo objeto de tipo `4D.File`. Es idéntico al comando `File` (acceso directo).

Se recomienda utilizar el comando de acceso directo `File<` en lugar de `4D.File.new()`.

## `.copyTo()`

► Histórico

`.copyTo( destinationFolder : 4D.Folder { ; newName : Text } { ; overwrite : Integer } ) : 4D.File`

Parámetros	Tipo		Descripción
destinationFolder	4D.Folder	->	Carpeta de destino
newName	Texto	->	Nombre para la copia
overwrite	Integer	->	<code>fk overwrite</code> para sustituir los elementos existentes
Resultado	4D.File	<-	Archivo copiado

## Descripción

La función `.copyTo()` copia el objeto `File` en el *destinationFolder* especificado> .

La *destinationFolder* debe existir en el disco, de lo contrario se genera un error.

Por defecto, el archivo se copia con el nombre del archivo original. Si desea cambiar el nombre de la copia, pase el nuevo nombre en el parámetro *newName*. El nuevo nombre debe cumplir con las reglas de nomenclatura (por ejemplo, no debe contener caracteres como ":", "/", etc.), de lo contrario se devuelve un error.

Si ya existe un archivo con el mismo nombre en la *destinationFolder*, por defecto 4D genera un error. Puede pasar la constante `fk overwrite` en el parámetro *overwrite* para ignorar y sobrescribir el archivo existente

Constante	Valor	Comentario
<code>fk overwrite</code>	4	Sobrescribir los elementos existentes, si los hay

## Valor devuelto

El objeto `File` copiado.

## Ejemplo

Desea copiar un *archivo* imagen de la carpeta de documentos del usuario a la carpeta de la aplicación:

```
var $source; $copy : Object
$source:=Folder(fk documents folder).file("Pictures/photo.png")
$copy:=$source.copyTo(Folder("/PACKAGE");fk overwrite)
```

## `.create()`

► Histórico

No disponible para archivos ZIP

`.create() : Boolean`

Parámetros	Tipo		Descripción
Resultado	Booleano	<-	True si el archivo se ha creado con éxito, false en caso contrario

## Descripción

The `.create()` function creates a file on disk according to the properties of the `File` object.

If necessary, the function creates the folder hierarchy as described in the `platformPath` or `path` properties. If the file already exists on disk, the function does nothing (no error is thrown) and returns false.

## Valor devuelto

- True si el archivo se crea con éxito;
- False si ya existe un archivo con el mismo nombre o si ha ocurrido un error.

## Ejemplo

Creation of a preferences file in the database folder:

```
var $created : Boolean
$created:=File("/PACKAGE/SpecialPrefs/"+Current user+".myPrefs").create()
```

## .createAlias()

### ► Histórico

`.createAlias( destinationFolder : 4D.Folder ; aliasName : Text { ; aliasType : Integer } ) : 4D.File`

Parámetros	Tipo		Descripción
destinationFolder	4D.Folder	->	Carpeta de destino para el alias o el acceso directo
aliasName	Texto	->	Name of the alias or shortcut
aliasType	Integer	->	Tipo de enlace del alias
Resultado	4D.File	<-	Referencia del archivo del alias o de atajo

## Descripción

The `.createAlias()` function creates an alias (macOS) or a shortcut (Windows) to the file with the specified `aliasName` name in the folder designated by the `destinationFolder` object.

Pass the name of the alias or shortcut to create in the `aliasName` parameter.

By default on macOS, the function creates a standard alias. You can also create a symbolic link by using the `aliasType` parameter. Las siguientes constantes están disponibles:

Constante	Valor	Comentario
<code>fk alias link</code>	0	Enlace de alias (por defecto)
<code>fk symbolic link</code>	1	Enlace simbólico (sólo para macOS)

On Windows, a shortcut (.lnk file) is always created (the `aliasType` parameter is ignored).

## Objeto devuelto

A `4D.File` object with the `isAlias` property set to true.

## Ejemplo

You want to create an alias to a file in your database folder:

```
$myFile:=Folder(fk documents folder).file("Archives/ReadMe.txt")
$aliasFile:=$myFile.createAlias(File("/PACKAGE");"ReadMe")
```

## .creationDate

► Histórico

.creationDate : Date

### Descripción

La propiedad `.creationDate` devuelve la fecha de creación del archivo.

Esta propiedad es de sólo lectura.

## .creationTime

► Histórico

.creationTime : Time

### Descripción

La propiedad `.creationTime` devuelve la hora de creación del archivo (expresada como un número de segundos que comienza en 00:00).

Esta propiedad es de sólo lectura.

## .delete()

► Histórico

.delete( )

Parámetros	Tipo	Descripción	-----	----	-----	-----		No requiere ningún parámetro
------------	------	-------------	-------	------	-------	-------	--	------------------------------

### Descripción

The `.delete()` function deletes the file.

If the file is currently open, an error is generated.

If the file does not exist on disk, the function does nothing (no error is generated).

**ATENCIÓN:** `.delete( )` puede eliminar cualquier archivo de un disco. Esto incluye los documentos creados con otras aplicaciones, así como las propias aplicaciones. `.delete( )` debe utilizarse con extrema precaución. Eliminar un archivo es una operación permanente y no se puede deshacer.

### Ejemplo

You want to delete a specific file in the database folder:

```
$tempo:=File("/PACKAGE/SpecialPrefs/"+Current user+".prefs")
If($tempo.exists)
    $tempo.delete()
    ALERT("Archivo de preferencias del usuario borrado.")
End if
End if
```

## .exists

► Histórico

.exists : Boolean

### Descripción

La propiedad `.exists` devuelve true si el archivo existe en el disco, y false en caso contrario.

Esta propiedad es de sólo lectura.

## .extension

► Histórico

.extension : Text

### Descripción

La propiedad `.extension` devuelve la extensión del nombre del archivo (si lo hay). Una extensión siempre comienza por `".`. La propiedad devuelve una cadena vacía si el nombre del archivo no tiene extensión.

Esta propiedad es de sólo lectura.

## .fullName

► Histórico

.fullName : Text

### Descripción

La propiedad `.fullName` devuelve el nombre completo del archivo, incluyendo su extensión (si la hay).

Esta propiedad es de sólo lectura.

## .getAppInfo()

► Histórico

.getAppInfo() : Object

Parámetros	Tipo		Descripción
Resultado	Objeto	<-	Contents of .exe/.dll version resource or .plist file

### Descripción

The `.getAppInfo()` function returns the contents of a `.exe`, `.dll` or `.plist` file information as an object.

The function must be used with an existing `.exe`, `.dll` or `.plist` file. If the file does not exist on disk or is not a valid `.exe`, `.dll` or `.plist` file, the function returns an empty object (no error is generated).

La función sólo admite archivos `.plist` en formato xml (basados en texto). Se devuelve un error si se utiliza con un archivo `.plist` en formato binario.

Objeto devuelto con un archivo `.exe` o `.dll`

La lectura de un `.exe` o `.dll` sólo es posible en Windows.

Todos los valores de propiedades son de tipo Texto.

Propiedad	Tipo
InternalName	Texto
ProductName	Texto
CompanyName	Texto
LegalCopyright	Texto
ProductVersion	Texto
FileDescription	Texto
FileVersion	Texto
OriginalFilename	Texto

Objeto devuelto con un archivo .plist

The xml file contents is parsed and keys are returned as properties of the object, preserving their types (text, boolean, number). `.plist dict` is returned as a JSON object and `.plist array` is returned as a JSON array.

## Ejemplo

```
// display copyright info of application .exe file (windows)
var $exeFile : 4D.File
var $info : Object
$exeFile:=File(Application file; fk platform path)
$info:=$exeFile.getAppInfo()
ALERT($info.LegalCopyright)

// display copyright info of an info.plist (any platform)
var $infoPlistFile : 4D.File
var $info : Object
$infoPlistFile:=File("/RESOURCES/info.plist")
$info:=$infoPlistFile.getAppInfo()
ALERT($info.Copyright)
```

Ver también

[.setAppInfo\(\)](#)

## .getContent()

► Histórico

`.getContent( )` : 4D.Blob

Parámetros	Tipo		Descripción
Resultado	4D.Blob	<-	Contenido del archivo

### Descripción

La función `.getContent()` devuelve un objeto `4D.Blob` que contiene todo el contenido de un archivo. Para obtener información sobre los BLOB, consulte la sección [BLOB](#).

Valor devuelto

Un objeto `4D.Blob`.

## Ejemplo

Para guardar el contenido de un documento en un campo `BLOB`:

```

var $vPath : Text
$vPath:=Select document("");*"Select a document";0)
If(OK=1) //Si se ha seleccionado un documento
[aTable]aBlobField:=File($vPath;fk platform path).getContent()
End if

```

## .getIcon()

► Histórico

.getIcon( { size : Integer } ) : Picture

Parámetros	Tipo		Descripción
size	Integer	->	Longitud del lado de la imagen devuelta (píxeles)
Resultado	Imagen	<-	Icono

### Descripción

La función `.getIcon()` devuelve el ícono del archivo.

El parámetro opcional `size` especifica las dimensiones en píxeles del ícono devuelto. Este valor representa en realidad la longitud del lado del cuadrado que contiene el ícono. Los íconos suelen definirse en 32x32 píxeles ("íconos grandes") o 16x16 píxeles ("íconos pequeños"). Si pasa 0 u omite este parámetro, se devuelve la versión "ícono grande".

Si el archivo no existe en el disco, se devuelve un ícono vacío por defecto.

### Valor devuelto

Ícono de archivo [picture](#).

## .getText()

► Histórico

.getText( { charSetName : Text { ; breakMode : Integer } } ) : Text  
.getText( { charSetNum : Integer { ; breakMode : Integer } } ) : Text

Parámetros	Tipo		Descripción
charSetName	Texto	->	Nombre del juego de caracteres
charSetNum	Integer	->	Número del conjunto de caracteres
breakMode	Integer	->	Modo de tratamiento de los saltos de línea
Resultado	Texto	<-	Texto del documento

### Descripción

La función `.getText()` devuelve el contenido del archivo como texto .

Opcionalmente, puede designar el conjunto de caracteres que se utilizará para leer el contenido. Puede pasar:

- en `charSetName`, una cadena que contiene el nombre del conjunto estándar (por ejemplo "ISO-8859-1" o "UTF-8"),
- o en `charSetNum`, el ID MIBEnum (número) del nombre del conjunto estándar.

Para conocer la lista de los conjuntos de caracteres que soporta 4D, consulte la descripción del comando [CONVERT FROM TEXT](#) .

Si el documento contiene una marca de orden de bytes (BOM), 4D utiliza el conjunto de caracteres que ha establecido en lugar del especificado en `charSetName` o `charSetNum` (este parámetro se ignora entonces). Si el documento no contiene una BOM y si `charSetName` o `charSetNum` se omite, por defecto 4D utiliza el conjunto de caracteres "UTF-8".

En *breakMode*, se puede pasar un número que indica el procesamiento a aplicar a los caracteres de fin de línea en el documento. Las siguientes constantes del tema "Documentos del sistema" están disponibles:

Constante	Valor	Comentario
Document unchanged	0	Sin procesar
Document with native format	1	(Por defecto) Los saltos de línea se convierten al formato nativo del sistema operativo: CR (retorno de carro) en OS X, CRLF (retorno de carro + salto de línea) en Windows
Document with CRLF	2	Los saltos de línea se convierten al formato de Windows: CRLF (retorno de carro + salto de línea)
Document with CR	3	Los saltos de línea se convierten al formato OS X: CR (retorno de carro)
Document with LF	4	Los saltos de línea se convierten al formato Unix: LF (salto de línea)

Por defecto, cuando se omite el parámetro *breakMode*, los saltos de línea se procesan en modo nativo (1).

Valor devuelto

Texto del archivo.

## Ejemplo

Dado el siguiente documento de texto (los campos están separados por tabulaciones):

```
id name price vat
3 thé 1.06€ 19.6
2 café 1.05€ 19.6
```

Cuando se ejecuta este código:

```
$myFile:=Folder(fk documents folder).file("Billing.txt") //UTF-8 por defecto
$txt:=$myFile.getText()
```

... obtiene para `$txt` :

"id\tname\tprice\tvat\r\n3\tthé\t1.06€\t19.6\r\n2\tcafé\t1.05€\t19.6"

con `\t` (tab) como separador y `\r\n` (CRLF) como delimitador de línea.

Aquí hay otro ejemplo con el mismo archivo, pero con un delimitador de línea diferente:

```
$txt:=$myFile.getText("UTF-8", Document with LF)
```

En este caso, el contenido de `$txt` es el siguiente:

"id\tname\tprice\tvat\n3\tthé\t1.06€\t19.6\n2\tcafé\t1.05€\t19.6"

Esta vez se utiliza `\n` (LF) como delimitador de línea.

## .hidden

► Histórico

.hidden : Boolean

Descripción

La propiedad `.hidden` devuelve true si el archivo está configurado como "oculto" a nivel de sistema, y false en caso contrario.

Esta propiedad es de sólo lectura.

## .isAlias

► Histórico

`.isAlias : Boolean`

### Descripción

La propiedad `.isAlias` devuelve true si el archivo es un alias, un acceso directo o un enlace simbólico, y false en caso contrario.

Esta propiedad es de sólo lectura.

## .isFile

► Histórico

`..isFile : Boolean`

### Descripción

La propiedad `.isFile` devuelve siempre verdadero para un archivo.

Esta propiedad es de sólo lectura.

## .isFolder

► Histórico

`.isFolder : Boolean`

### Descripción

La propiedad `.isFolder` devuelve siempre false para un archivo.

Esta propiedad es de sólo lectura.

## .isWritable

► Histórico

`.isWritable : Boolean`

### Descripción

La propiedad `.isWritable` devuelve true si el archivo existe en el disco y es escribible.

La propiedad verifica la capacidad de la aplicación 4D a escribir en el disco (derechos de acceso), no se basa únicamente en el atributo *writable* del archivo.

Esta propiedad es de sólo lectura.

### Ejemplo

```

$myFile:=File("C:\\Documents\\Archives\\ReadMe.txt";fk platform path)
If($myFile.isWritable)
    $myNewFile:=$myFile.setText("Added text")
End if

```

## .modificationDate

► Histórico

.modificationDate : Date

### Descripción

La propiedad `.modificationDate` devuelve la fecha de la última modificación del archivo.

Esta propiedad es de sólo lectura.

## .modificationTime

► Histórico

.modificationTime : Time

### Descripción

La propiedad `.modificationTime` devuelve la hora de la última modificación del archivo (expresada como un número de segundos que comienza en 00:00).

Esta propiedad es de sólo lectura.

## .moveTo()

► Histórico

.moveTo( *destinationFolder* : 4D.Folder { ; *newName* : Text } ) : 4D.File

Parámetros	Tipo		Descripción
<i>destinationFolder</i>	4D.Folder	->	Carpeta de destino
<i>newName</i>	Texto	->	Nombre completo del archivo trasladado
Resultado	4D.File	<-	Archivo movido

### Descripción

The `.moveTo()` function moves or renames the `File` object into the specified *destinationFolder*.

La *destinationFolder* debe existir en el disco, de lo contrario se genera un error.

Por defecto, el archivo conserva su nombre cuando se mueve. If you want to rename the moved file, pass the new full name in the *newName* parameter. El nuevo nombre debe cumplir con las reglas de nomenclatura (por ejemplo, no debe contener caracteres como ":" , "/" , etc.), de lo contrario se devuelve un error.

### Objeto devuelto

El objeto `File` movido.

### Ejemplo

```

$DocFolder:=Folder(fk documents folder)
$myFile:=$DocFolder.file("Current/Infos.txt")
$myFile.moveTo($DocFolder.folder("Archives");"Infos_old.txt")

```

## .name

► Histórico

.name : Text

### Descripción

La propiedad `.name` devuelve el nombre del archivo, sin extensión (si la hay).

Esta propiedad es de sólo lectura.

## .original

► Histórico

.original : 4D.File

.original : 4D.Folder

### Descripción

La propiedad `.original` devuelve el elemento de destino para un alias, un acceso directo o un archivo de enlace simbólico. El elemento objetivo puede ser:

- un objeto File
- un objeto de la carpeta

Para los archivos sin alias, la propiedad devuelve el mismo objeto File que el archivo.

Esta propiedad es de sólo lectura.

## .parent

► Histórico

.parent : 4D.Folder

### Descripción

La propiedad `.parent` devuelve el objeto carpeta padre del archivo. Si la ruta representa una ruta del sistema (por ejemplo, "/DATA/"), se devuelve la ruta del sistema.

Esta propiedad es de sólo lectura.

## .path

► Histórico

.path : Text

### Descripción

La propiedad `.path` devuelve la ruta POSIX del archivo. Si la ruta representa un filesystem (por ejemplo, "/DATA/"), se devuelve el filesystem.

Esta propiedad es de sólo lectura.

## .platformPath

► Histórico

.platformPath : Text

### Descripción

La propiedad `.platformPath` devuelve la ruta del archivo expresada con la sintaxis de la plataforma actual.

Esta propiedad es de sólo lectura.

## .rename()

► Histórico

.rename( *newName* : Text ) : 4D.File

Parámetros	Tipo		Descripción
<i>newName</i>	Texto	->	Nuevo nombre completo para el archivo
Resultado	4D.File	<-	Archivo renombrado

### Descripción

The `.rename()` function renames the file with the name you passed in *newName* and returns the renamed `File` object.

The *newName* parameter must comply with naming rules (e.g., it must not contain characters such as ":", "/", etc.), otherwise an error is returned. If a file with the same name already exists, an error is returned.

Note that the function modifies the full name of the file, i.e. if you do not pass an extension in *newName*, the file will have a name without an extension.

### Objeto devuelto

El objeto `File` renombrado.

### Ejemplo

You want to rename "ReadMe.txt" in "ReadMe\_new.txt":

```
$toRename:=File("C:\\Documents\\Archives\\ReadMe.txt";fk platform path)
$newName:=$toRename.rename($toRename.name+"_new"+$toRename.extension)
```

## .setAppInfo()

► Histórico

.setAppInfo( *info* : Object )

Parámetros	Tipo		Descripción
<i>info</i>	Objeto	->	Properties to write in .exe/.dll version resource or .plist file

### Descripción

The `.setAppInfo()` function writes the *info* properties as information contents of a .exe, .dll or .plist file.

The function must be used with an existing .exe, .dll or .plist file. The function must be used with an existing .exe, .dll or .plist file.

La función sólo admite archivos .plist en formato xml (basados en texto). Se devuelve un error si se utiliza con un archivo .plist en formato binario.

*info* parameter object with a .exe or .dll file

Writing a .exe or .dll file information is only possible on Windows.

Each valid property set in the *info* object parameter is written in the version resource of the .exe or .dll file. Available properties are (any other property will be ignored):

Propiedad	Tipo
InternalName	Texto
ProductName	Texto
CompanyName	Texto
LegalCopyright	Texto
ProductVersion	Texto
FileDescription	Texto
FileVersion	Texto
OriginalFilename	Texto

If you pass a null or empty text as value, an empty string is written in the property. If you pass a value type different from text, it is stringified.

*info* parameter object with a .plist file

Each valid property set in the *info* object parameter is written in the .plist file as a key. Se acepta todo nombre de llave. Los tipos de valores se conservan cuando es posible.

If a key set in the *info* parameter is already defined in the .plist file, its value is updated while keeping its original type. Other existing keys in the .plist file are left untouched.

To define a Date type value, the format to use is a json timestamp string formated in ISO UTC without milliseconds ("2003-02-01T01:02:03Z") like in the Xcode plist editor.

## Ejemplo

```
// set copyright and version of a .exe file (Windows)
var $exeFile : 4D.File
var $info : Object
$exeFile:=File(Application file; fk platform path)
$info:=New object
$info.LegalCopyright:="Copyright 4D 2021"
$info.ProductVersion:="1.0.0"
$exeFile.setAppInfo($info)
```

```
// set some keys in an info.plist file (all platforms)
var $infoPlistFile : 4D.File
var $info : Object
$infoPlistFile:=File("/RESOURCES/info.plist")
$info:=New object
$info.Copyright:="Copyright 4D 2021" //text
$info.ProductVersion:=12 //integer
$info.ShipmentDate:="2021-04-22T06:00:00Z" //timestamp
$infoPlistFile.setAppInfo($info)
```

Ver también

[.getAppInfo\(\)](#)

## .setContent()

► Histórico

[.setContent \( content : Blob \)](#)

Parámetros	Tipo		Descripción
content	BLOB	->	Nuevos contenidos para el archivo

## Descripción

The `.setContent( )` function rewrites the entire content of the file using the data stored in the `content` BLOB. Para obtener información sobre los BLOB, consulte la sección [BLOB](#).

## Ejemplo

```
$myFile:=Folder(fk documents folder).file("Archives/data.txt")
$myFile.setContent([aTable]aBlobField)
```

## .setText()

### ► Histórico

`.setText ( text : Text {; charSetName : Text { ; breakMode : Integer } } )`  
`.setText ( text : Text {; charSetNum : Integer { ; breakMode : Integer } } )`

Parámetros	Tipo		Descripción
texto	Texto	->	Texto a almacenar en el archivo
charSetName	Texto	->	Nombre del juego de caracteres
charSetNum	Integer	->	Número del conjunto de caracteres
breakMode	Integer	->	Modo de tratamiento de los saltos de línea

## Descripción

The `.setText()` function writes `text` as the new contents of the file.

If the file referenced in the `File` object does not exist on the disk, it is created by the function. When the file already exists on the disk, its prior contents are erased, except if it is already open, in which case, its contents are locked and an error is generated.

In `text`, pass the text to write to the file. It can be a literal ("my text"), or a 4D text field or variable.

Optionally, you can designate the character set to be used for writing the contents. Puede pasar:

- en `charSetName`, una cadena que contiene el nombre del conjunto estándar (por ejemplo "ISO-8859-1" o "UTF-8"),
- o en `charSetNum`, el ID MIBEnum (número) del nombre del conjunto estándar.

Para conocer la lista de los conjuntos de caracteres que soporta 4D, consulte la descripción del comando `CONVERT FROM TEXT`.

Si existe una marca de orden de bytes (BOM) para el conjunto de caracteres, 4D la inserta en el archivo a menos que el conjunto de caracteres utilizado contenga el sufijo "-no-bom" (por ejemplo, "UTF-8-no-bom"). Si no especifica un conjunto de caracteres, por defecto 4D utiliza el conjunto de caracteres "UTF-8" sin BOM.

In `breakMode`, you can pass a number indicating the processing to apply to end-of-line characters before saving them in the file. Las siguientes constantes, que se encuentran en el tema Documentos sistema, están disponibles:

Constante	Valor	Comentario
Document unchanged	0	Sin procesar
Document with native format	1	(Por defecto) Los saltos de línea se convierten al formato nativo del sistema operativo: LF (salto de línea) en macOS, CRLF (salto de línea + retorno de carro) en Windows
Document with CRLF	2	Los saltos de línea se convierten en CRLF (retorno de carro + salto de línea), el formato predeterminado de Windows
Document with CR	3	Los saltos de línea se convierten en CR (retorno de carro), el formato clásico por defecto de Mac OS
Document with LF	4	Los saltos de línea se convierten en LF (salto de línea), el formato por defecto de Unix y macOS

Por defecto, cuando se omite el parámetro *breakMode*, los saltos de línea se procesan en modo nativo (1).

Nota de compatibilidad: las opciones de compatibilidad están disponibles para la gestión de EOL y de BOM.  
Ver [Página Compatibilidad](#) en doc.4d.com.

## Ejemplo

```
$myFile:=File("C:\\\\Documents\\\\Hello.txt";fk platform path)
$myFile.setText("Hello world")
```

## .size

► Histórico

.size : Real

### Descripción

La propiedad `.size` devuelve el tamaño del archivo expresado en bytes. Si el archivo no existe en el disco, el tamaño es 0.

Esta propiedad es de sólo lectura.

# Folder

Los objetos `Folder` se crean con el comando `Folder`. Contienen referencias a carpetas que pueden o no existir realmente en el disco. Por ejemplo, cuando se ejecuta el comando `Folder` para crear una nueva carpeta, se crea un objeto `Folder` válido, pero en realidad no se almacena nada en el disco hasta que se llama a la función `folder.create()`.

## Ejemplo

El siguiente ejemplo crea una carpeta "JohnSmith":

```
Form.curfolder:=Folder(fk database folder)
Form.curfolder:=Folder("C:\\\\Users\\\\JohnSmith\\\\";fk platform path)
```

## Objeto Folder

`.copyTo( destinationFolder : 4D.Folder { ; newName : Text } { ; overwrite : Integer } ) : 4D Folder`  
copia el objeto `Folder` en el `destinationFolder` especificado>

`.create() : Boolean`  
crea una carpeta en el disco según las propiedades del objeto `Folder`

`.createAlias( destinationFolder : 4D.Folder ; aliasName : Text { ; aliasType : Integer } ) : 4D.File`  
crea un alias (macOS) o un acceso directo (Windows)

`.creationDate : Date`  
la fecha de creación de la carpeta

`.creationTime : Time`  
la hora de creación de la carpeta

`.delete( { option : Integer } )`  
borra el archivo

`.exists : Boolean`  
true si la carpeta existe en el disco

`.extension : Text`  
devuelve la extensión del nombre de la carpeta (si la hay)

`.file( path : Text ) : 4D.File`  
crea un objeto `File` en el objeto `Folder` y devuelve su referencia

`.files( { options : Integer } ) : Collection`  
una colección de objetos `File` contenidos en la carpeta

`.folder( path : Text ) : 4D.Folder`  
crea un objeto `Folder` dentro del objeto padre `Folder` y devuelve su referencia

`.folders( { options : Integer } ) : Collection`  
devuelve una colección de objetos `Folder` contenidos en la carpeta padre

<code>.fullName : Text</code>	devuelve el nombre completo de la carpeta, incluyendo su extensión (si la hay)
<code>.getIcon( { size : Integer } ) : Picture</code>	devuelve el ícono de la carpeta
<code>.hidden : Boolean</code>	true si la carpeta está configurada como "oculta" a nivel de sistema
<code>.isAlias : Boolean</code>	siempre false para un objeto <code>Folder</code>
<code>.isFile : Boolean</code>	siempre false para una carpeta
<code>.isFolder : Boolean</code>	siempre true para una carpeta
<code>.isPackage : Boolean</code>	true si la carpeta es un paquete en macOS (y existe en el disco)
<code>.modificationDate : Date</code>	la fecha de la última modificación de la carpeta
<code>.modificationTime : Time</code>	la hora de la última modificación de la carpeta
<code>.name : Text</code>	el nombre de la carpeta, sin extensión (si la hay)
<code>.original : 4D.Folder</code>	el mismo objeto <code>Folder</code> que la carpeta
<code>.parent : 4D.Folder</code>	el objeto carpeta padre de la carpeta
<code>.path : Text</code>	la ruta POSIX de la carpeta
<code>.platformPath : Text</code>	la ruta de la carpeta expresada con la sintaxis de la plataforma actual
<code>.moveTo( destinationFolder : 4D.Folder { ; newName : Text } ) : 4D.Folder</code>	mueve o renombra el objeto <code>Folder</code> (carpeta fuente) a la <i>carpeta de destino</i>
<code>.rename( newName : Text ) : 4D.Folder</code>	renombra el archivo con el nombre que se ha pasado en <code>newName</code> y devuelve el objeto <code>Folder</code> renombrado

## Folder

► Histórico

`Folder ( path : Text { ; pathType : Integer }{ ; * } ) : 4D.Folder`

`Folder ( folderConstant : Integer { ; * } ) : 4D.Folder`

Parámetros	Tipo		Descripción
path	Texto	->	Ruta de la carpeta
folderConstant	Integer	->	Constante de la carpeta 4D
pathType	Integer	->	<code>fk posix path</code> (por defecto) o <code>fk platform path</code>
*		->	* para devolver la carpeta de la base local
Resultado	4D.Folder	<-	Nuevo objeto de carpeta

## Descripción

El comando `Folder` crea y devuelve un nuevo objeto de tipo `4D.Folder`. El comando acepta dos sintaxis:

`Folder ( path { ; pathType } { ; * } )`

En el parámetro *path*, pase una ruta de carpeta. Puede utilizar una cadena personalizada o un filesystem (por ejemplo, `"/DATA"`).

Sólo se soportan los nombres de ruta absolutos con el comando `Folder`.

Por defecto, 4D espera una ruta expresada con la sintaxis POSIX. Si trabaja con los nombres de ruta de plataforma (Windows o macOS), debe declararlo utilizando el parámetro *pathType*. Las siguientes constantes están disponibles:

Constante	Valor	Comentario
<code>fk platform path</code>	1	Ruta expresada con una sintaxis específica de la plataforma (obligatoria en caso de nombre de ruta de plataforma)
<code>fk posix path</code>	0	Ruta expresada con sintaxis POSIX (por defecto)

`Folder ( folderConstant { ; * } )`

En el parámetro *folderConstant*, pase una carpeta 4D interna o sistema, utilizando una de las siguientes constantes:

Constante	Valor	Comentario
fk applications folder	116	
fk data folder	9	Filesystem asociado: "/DATA"
fk database folder	4	Filesystem asociado: "/PACKAGE"
fk desktop folder	115	
fk documents folder	117	Carpeta Documentos del usuario
fk licenses folder	1	Carpeta que contiene los archivos de licencia 4D de la máquina
fk logs folder	7	Filesystem asociado: "/LOGS"
fk mobileApps folder	10	
fk remote database folder	3	Carpeta de la base de datos 4D creada en cada máquina 4D remota
fk resources folder	6	Filesystem asociado: "/RESOURCES"
fk system folder	100	
fk user preferences folder	0	Carpeta 4D que almacena los archivos de preferencias del usuario en el <code>¥N&lt;username&gt;</code> directorio.
fk web root folder	8	Carpeta raíz web actual de la base de datos: si está dentro del paquete "/PACKAGE/path", si no ruta completa

Si el comando se llama desde un componente, pase el parámetro opcional \* para obtener la ruta de la base local. De lo contrario, si omite el parámetro \*, siempre se devuelve un objeto null.

## 4D.Folder.new()

► Histórico

`4D.Folder.new ( path : Text { ; pathType : Integer }{ ; * } ) : 4D.Folder`

`4D.Folder.new ( folderConstant : Integer { ; * } ) : 4D.Folder`

### Descripción

La función `4D.Folder.new()` crea y devuelve un nuevo objeto de tipo `4D.Folder`. Es idéntico al comando `Folder` (acceso directo).

Se recomienda utilizar el comando de acceso directo `Folder<` en lugar de `4D.Folder.new()`.

## .copyTo()

► Histórico

`.copyTo( destinationFolder : 4D.Folder { ; newName : Text } { ; overwrite : Integer } ) : 4D Folder`

Parámetros	Tipo		Descripción
destinationFolder	4D.Folder	->	Carpeta de destino
newName	Texto	->	Nombre para la copia
overwrite	Integer	->	<code>fk overwrite</code> para sustituir los elementos existentes
Resultado	4D.Folder	<-	Carpeta o archivo copiado

## Descripción

La función `.copyTo()` copia el objeto `Folder` en el *destinationFolder* especificado>.

La *destinationFolder* debe existir en el disco, de lo contrario se genera un error.

Por defecto, la carpeta se copia con el nombre de la carpeta original. Si desea cambiar el nombre de la copia, pase el nuevo nombre en el parámetro *newName*. El nuevo nombre debe cumplir con las reglas de nomenclatura (por ejemplo, no debe contener caracteres como ":" "/", etc.), de lo contrario se devuelve un error.

Si ya existe una carpeta con el mismo nombre en la *destinationFolder*, por defecto 4D genera un error. Puede pasar la constante `fk overwrite` en el parámetro *overwrite* para ignorar y sobrescribir el archivo existente

Constante	Valor	Comentario
<code>fk overwrite</code>	4	Sobrescribir los elementos existentes, si los hay

Valor devuelto

El objeto `Folder` copiado.

## Ejemplo

Desea copiar una *carpeta Imágenes* de la carpeta de documentos del usuario a la carpeta de la base:

```
var $userImages; $copiedImages : 4D.Folder  
$userImages:=Folder(fk documents folder+"/Pictures/")  
$copiedImages:=$userImages.copyTo(Folder(fk database folder);fk overwrite)
```

## .create()

► Histórico

`.create() : Boolean`

Parámetros	Tipo		Descripción
Resultado	Booleano	<-	True si la carpeta se ha creado con éxito, false en caso contrario

## Descripción

La función `.create()` crea una carpeta en el disco según las propiedades del objeto `Folder`.

Si es necesario, la función crea la jerarquía de carpetas como se describe en las propiedades `platformPath` o `path`. Si la carpeta ya existe en el disco, la función no hace nada (no se lanza ningún error) y devuelve false.

Valor devuelto

- True si la carpeta se crea con éxito;
- False si ya existe una carpeta con el mismo nombre o si ha ocurrido un error.

## Ejemplo 1

Cree una carpeta vacía en la carpeta de la base:

```
var $created : Boolean  
$created:=Folder("/PACKAGE/SpecialPrefs").create()
```

## Ejemplo 2

Creación de la carpeta "/Archives2019/January/" en la carpeta de la base:

```

$newFolder:=Folder("/PACKAGE/Archives2019/January")
If($newFolder.create())
    ALERT("La"+$newFolder.name+" carpeta fue creada.")
Else
    ALERT("Impossible to create a "+$newFolder.name+" folder.")
End if
Else
    ALERT("Impossible to create a "+$newFolder.name+" folder.")
End if

```

## .createAlias()

► Histórico

.createAlias( *destinationFolder* : 4D.Folder ; *aliasName* : Text { ; *aliasType* : Integer } ) : 4D.File

Parámetros	Tipo		Descripción
<i>destinationFolder</i>	4D.Folder	->	Carpeta de destino para el alias o el acceso directo
<i>aliasName</i>	Texto	->	Nombre del alias o del atajo
<i>aliasType</i>	Integer	->	Tipo de enlace del alias
Resultado	4D.File	<-	Referencia de alias o de acceso directo

### Descripción

La función `.createAlias()` crea un alias (macOS) o un acceso directo (Windows) para la carpeta con el nombre *aliasName* especificado en la carpeta designada por el objeto *destinationFolder*.

Pase el nombre del alias o del acceso directo a crear en el parámetro *aliasName*.

Por defecto en macOS, la función crea un alias estándar. También puede crear un enlace simbólico utilizando el parámetro *aliasType*. Las siguientes constantes están disponibles:

Constante	Valor	Comentario
<code>fk alias link</code>	0	Enlace de alias (por defecto)
<code>fk symbolic link</code>	1	Enlace simbólico (sólo para macOS)

En Windows, siempre se crea un acceso directo (archivo.lnk) (el parámetro *aliasType* es ignorado).

### Objeto devuelto

Un objeto `4D.File` con la propiedad `isAlias` definida en true.

### Ejemplo

Quiere crear un alias para una carpeta de archivos en su carpeta de base:

```

$myFolder:=Folder("C:\\\\Documents\\\\Archives\\\\2019\\\\January";fk platform path)
$aliasFile:=$myFolder.createAlias(Folder("/PACKAGE");"Jan2019")

```

## .creationDate

► Histórico

.creationDate : Date

### Descripción

La propiedad `.creationDate` devuelve la fecha de creación de la carpeta.

Esta propiedad es de sólo lectura.

## .creationTime

► Histórico

`.creationTime : Time`

### Descripción

La propiedad `.creationTime` devuelve la hora de creación de la carpeta (expresada como un número de segundos que comienza en 00:00).

Esta propiedad es de sólo lectura.

## .delete()

► Histórico

`.delete( { option : Integer } )`

Parámetros	Tipo		Descripción
option	Integer	->	Opción de eliminación de carpeta

### Descripción

La función `.delete()` borra el archivo.

Por defecto, por razones de seguridad, si se omite el parámetro `option`, `.delete( )` sólo permite borrar las carpetas vacías. Si desea que el comando pueda eliminar carpetas que no están vacías, debe utilizar el parámetro `option` con una de las siguientes constantes:

Constante	Valor	Comentario
<code>Delete only if empty</code>	0	Elimina la carpeta sólo cuando está vacía
<code>Delete with contents</code>	1	Elimina la carpeta junto con todo lo que contiene

Cuando `Delete only if empty` se pasa o si se omite el parámetro `option`:

- La carpeta sólo se elimina si está vacía; en caso contrario, el comando no hace nada y se genera un error -47.
- Si la carpeta no existe, se genera el error -120.

Cuando se pasa `Delete with contents` :

- La carpeta, junto con todo su contenido, se elimina. Advertencia: incluso esta carpeta y/o su contenido estén bloqueados o definidos como de sólo lectura, si el usuario actual tiene los derechos de acceso adecuados, la carpeta (y su contenido) aún se elimina.
- Si esta carpeta, o cualquiera de los archivos que contiene, no puede ser eliminada, la eliminación se interrumpe tan pronto como se detecta el primer elemento inaccesible y se devuelve un error(\*). En este caso, la carpeta puede ser eliminada sólo parcialmente. Cuando se interrumpe el borrado, puede utilizar el comando `GET LAST ERROR STACK` para recuperar el nombre y la ruta de acceso del archivo infractor.
- Si la carpeta no existe, el comando no hace nada y no devuelve ningún error. (\*) Windows: -54 (Intento de abrir un archivo bloqueado para escribir) macOS: -45 (El archivo está bloqueado o la ruta no es correcta)

## .exists

► Histórico

`.exists : Boolean`

### Descripción

La propiedad `.exists` devuelve true si la carpeta existe en el disco, y false en caso contrario.

Esta propiedad es de sólo lectura.

## .extension

► Histórico

`.extension` : Text

### Descripción

La propiedad `.extension` devuelve la extensión del nombre de la carpeta (si la hay). Una extensión siempre comienza por `".`. La propiedad devuelve una cadena vacía si el nombre de la carpeta no tiene extensión.

Esta propiedad es de sólo lectura.

## .file()

► Histórico

`.file( path : Text )` : 4D.File

Parámetros	Tipo		Descripción
path	Texto	->	Ruta POSIX relativa
Resultado	4D.File	<-	Objeto <code>File</code> (null si la ruta no es válida)

### Descripción

La función `.file()` crea un objeto `File` en el objeto `Folder` y devuelve su referencia.

En `path`, pase una ruta relativa POSIX para designar el archivo a devolver. La ruta se evaluará a partir de la carpeta padre como raíz.

### Valor devuelto

Un objeto `File` o null si `path` no es válido.

### Ejemplo

```
var $myPDF : 4D.File  
$myPDF:=Folder(fk documents folder).file("Pictures/info.pdf")
```

## .files()

► Histórico

`.files( { options : Integer } )` : Collection

Parámetros	Tipo		Descripción
options	Integer	->	Opciones de la lista de archivos
Resultado	Collection	<-	Colección de objetos de archivo hijo

### Descripción

La función `.files()` devuelve una colección de objetos `File` contenidos en la carpeta.

Los alias o enlaces simbólicos no se resuelven.

Por defecto, si se omite el parámetro *options*, sólo se devuelven en la colección los archivos del primer nivel de la carpeta, así como los archivos o carpetas invisibles. Puede modificar esto pasando, en el parámetro *options*, una o varias de las siguientes constantes:

Constante	Valor	Comentario
fk recursive	1	La colección contiene los archivos de la carpeta especificada y sus subcarpetas
fk ignore invisible	8	Los archivos invisibles no aparecen en la lista

Valor devuelto

Colección de objetos `File`.

### Ejemplo 1

Quiere saber si hay archivos invisibles en la carpeta de la base:

```
var $all; $noInvisible : Collection
$all:=Folder(fk database folder).files()
$noInvisible:=Folder(fk database folder).files(fk ignore invisible)
If($all.length#$noInvisible.length)
    ALERT("Database folder contains hidden files.")
End if
```

### Ejemplo 2

Quiere obtener todos los archivos que no son invisibles en la carpeta Documents:

```
var $recursive : Collection
$recursive:=Folder(fk documents folder).files(fk recursive+fk ignore invisible)
```

## .folder()

► Histórico

`.folder( path : Text ) : 4D.Folder`

Parámetros	Tipo		Descripción
path	Texto	->	Ruta POSIX relativa
Resultado	4D.Folder	<-	Objeto carpeta creado (null si <i>path</i> no es válido)

### Descripción

La función `.folder()` crea un objeto `Folder` dentro del objeto padre `Folder` y devuelve su referencia.

En *path*, pase una ruta relativa POSIX para designar la carpeta a devolver. La ruta se evaluará a partir de la carpeta padre como raíz.

Valor devuelto

Un objeto `Folder` o null si *path* no es válido.

### Ejemplo

```
var $mypicts : 4D.Folder
$mypicts:=Folder(fk documents folder).folder("Pictures")
```

## .folders()

► Histórico

.folders( { options : Integer } ) : Collection

Parámetros	Tipo		Descripción
options	Integer	->	Opciones de la lista de carpetas
Resultado	Collection	<-	Colección de objetos de carpeta hijo

### Descripción

La función `.folders()` devuelve una colección de objetos `Folder` contenidos en la carpeta padre.

Por defecto, si se omite el parámetro `options`, sólo se devuelven en la colección las carpetas del primer nivel de la carpeta. Puede modificar esto pasando, en el parámetro `options`, una o varias de las siguientes constantes:

Constante	Valor	Comentario
<code>fk recursive</code>	1	La colección contiene las carpetas de la carpeta especificada y sus subcarpetas
<code>fk ignore invisible</code>	8	Los archivos invisibles no aparecen en la lista

### Valor devuelto

Colección de objetos `Folder`.

### Ejemplo

Quiere obtener la colección de todas las carpetas y subcarpetas de la carpeta de la base:

```
var $allFolders : Collection  
$allFolders:=Folder("/PACKAGE").folders(fk recursive)
```

## .fullName

► Histórico

.fullName : Text

### Descripción

La propiedad `.fullName` devuelve el nombre completo de la carpeta, incluyendo su extensión (si la hay).

Esta propiedad es de sólo lectura.

## .getIcon()

► Histórico

.getIcon( { size : Integer } ) : Picture

Parámetros	Tipo		Descripción
size	Integer	->	Longitud del lado de la imagen devuelta (píxeles)
Resultado	Imagen	<-	Icono

### Descripción

La función `.getIcon()` devuelve el ícono de la carpeta.

El parámetro opcional `size` especifica las dimensiones en píxeles del ícono devuelto. Este valor representa en realidad la

longitud del lado del cuadrado que contiene el ícono. Los íconos suelen definirse en 32x32 píxeles ("íconos grandes") o 16x16 píxeles ("íconos pequeños"). Si pasa 0 u omite este parámetro, se devuelve la versión "ícono grande".

Si la carpeta no existe en el disco, se devuelve un ícono vacío por defecto.

Valor devuelto

[Imagen](#) del ícono de la carpeta.

## .hidden

► Histórico

.hidden : Boolean

Descripción

La propiedad `.hidden` devuelve true si la carpeta está configurada como "oculta" a nivel de sistema, y false en caso contrario.

Esta propiedad es de sólo lectura.

## .isAlias

► Histórico

.isAlias : Boolean

Descripción

La propiedad `.isAlias` devuelve siempre false para un objeto `Folder`.

Esta propiedad es de sólo lectura.

## .isFile

► Histórico

..isFile : Boolean

Descripción

La propiedad `.isFile` devuelve siempre false para una carpeta.

Esta propiedad es de sólo lectura.

## .isFolder

► Histórico

.isFolder : Boolean

Descripción

La propiedad `.isFolder` devuelve siempre true para una carpeta.

Esta propiedad es de sólo lectura.

## .isPackage

► Histórico

.isPackage : Boolean

Descripción

La propiedad `.isPackage` devuelve true si la carpeta es un paquete en macOS (y existe en el disco). En caso contrario, devuelve false.

En Windows, `.isPackage` siempre devuelve false.

Esta propiedad es de sólo lectura.

## .modificationDate

► Histórico

`.modificationDate : Date`

Descripción

La propiedad `.modificationDate` devuelve la fecha de la última modificación de la carpeta.

Esta propiedad es de sólo lectura.

## .modificationTime

► Histórico

`.modificationTime : Time`

Descripción

La propiedad `.modificationTime` devuelve la hora de la última modificación de la carpeta (expresada como un número de segundos que comienza en 00:00).

Esta propiedad es de sólo lectura.

## .moveTo()

► Histórico

`.moveTo( destinationFolder : 4D.Folder { ; newName : Text } ) : 4D.Folder`

Parámetros	Tipo		Descripción
destinationFolder	4D.Folder	->	Carpeta de destino
newName	Texto	->	Nombre completo de la carpeta trasladada
Resultado	4D.Folder	<-	Carpeta movida

Descripción

La función `.moveTo( )` mueve o renombra el objeto `Folder` (carpeta fuente) a la *carpeta de destino*.

La *destinationFolder* debe existir en el disco, de lo contrario se genera un error.

Por defecto, la carpeta conserva su nombre cuando se mueve. Si desea cambiar renombrar la carpeta desplazada, pase el nombre completo en el parámetro *newName*. El nuevo nombre debe cumplir con las reglas de nomenclatura (por ejemplo, no debe contener caracteres como ":"; "/", etc.), de lo contrario se devuelve un error.

Objeto devuelto

El objeto `Folder` movido.

Ejemplo

Quiere mover y renombrar una carpeta:

```
var $tomeove; $moved : Object  
$docs:=Folder(fk documents folder)  
$tomeove:=$docs.folder("Pictures")  
$tomeove2:=$tomeove.moveTo($docs.folder("Archives");"Pic_Archives")
```

## .name

► Histórico

.name : Text

### Descripción

La propiedad `.name` devuelve el nombre de la carpeta, sin extensión (si la hay).

Esta propiedad es de sólo lectura.

## .original

► Histórico

.original : 4D.Folder

### Descripción

La propiedad `.original` devuelve el mismo objeto Folder que la carpeta.

Esta propiedad es de sólo lectura.

Esta propiedad está disponible en las carpetas para permitir que el código genérico procese carpetas o archivos.

## .parent

► Histórico

.parent : 4D.Folder

### Descripción

La propiedad `.parent` devuelve el objeto carpeta padre de la carpeta. Si la ruta representa una ruta del sistema (por ejemplo, "/DATA/"), se devuelve la ruta del sistema.

Si la carpeta no tiene un parent (raíz), se devuelve el valor null.

Esta propiedad es de sólo lectura.

## .path

► Histórico

.path : Text

### Descripción

La propiedad `.path` devuelve la ruta POSIX de la carpeta. Si la ruta representa un filesystem (por ejemplo, "/DATA/"), se devuelve el filesystem.

Esta propiedad es de sólo lectura.

## .platformPath

► Histórico

.platformPath : Text

## Descripción

La propiedad `.platformPath` devuelve la ruta de la carpeta expresada con la sintaxis de la plataforma actual.

Esta propiedad es de sólo lectura.

## .rename()

► Histórico

.rename( *newName* : Text ) : 4D.Folder

Parámetros	Tipo		Descripción
<i>newName</i>	Texto	->	Nuevo nombre completo para la carpeta
Resultado	4D.Folder	<-	Carpeta renombrada

## Descripción

La función `.rename()` renombra el archivo con el nombre que se ha pasado en *newName* y devuelve el objeto `Folder` renombrado.

El parámetro *newName* debe cumplir con las reglas de nomenclatura (por ejemplo, no debe contener caracteres como ":"; "/", etc.), de lo contrario se devuelve un error. Si ya existe un archivo con el mismo nombre, se devuelve un error.

## Objeto devuelto

El objeto `Folder` renombrado.

## Ejemplo

```
var $toRename : 4D.Folder  
$toRename:=Folder("/RESOURCES/Pictures").rename("Images")
```

# Formula

Los comandos [Formula](#) y [Formula from string](#) le permiten crear los objetos [4D.Function](#) para ejecutar toda expresión o código 4D expresado como texto.

## Objetos Formula

Los objetos Formula pueden encapsularse en las propiedades de objeto:

```
var $f : 4D.Function  
$f:=New object  
$f.message:=Formula(ALERT("Hello world"))
```

Esta propiedad es una "función objeto", es decir una función que está vinculada a su objeto padre. Para ejecutar una función almacenada en una propiedad objeto, utilice el operador () después del nombre de la propiedad, como:

```
$f.message() //muestra "Hello world"
```

También se admite la sintaxis con paréntesis:

```
$f["message"]() //muestra "Hello world"
```

Tenga en cuenta que, aunque no tenga parámetros (ver arriba), una función objeto a ejecutar debe ser llamada con paréntesis (). Llamar sólo a la propiedad del objeto devolverá una nueva referencia a la fórmula (y no la ejecutará):

```
$o:=$f.message //devuelve el objeto fórmula en $o
```

Para ejecutar una función utilizando las funciones [apply\(\)](#) y [call\(\)](#):

```
$f.message.apply() //muestra "Hello world"
```

## Paso de parámetros

Puede pasar parámetros a sus fórmulas utilizando la [sintaxis secuencial de los parámetros](#) basada en \$1, \$2...\$n. Por ejemplo, puede escribir:

```
var $f : Object  
$f:=New object  
$f.message:=Formula(ALERT("Hello "+$1))  
$f.message("John") //muestra "Hello John"
```

O utilizando la función [.call\(\)](#):

```
var $f : Object  
$f:=Formula($1+" "+$2)  
$text:=$f.call(Null;"Hello";"World") //devuelve "Hello World"  
$text:=$f.call(Null;"Welcome to";String(Year of(Current date))) //devuelve "Welcome to 2019" (por ejemp
```

## Parámetros de un solo método

Para mayor comodidad, cuando la fórmula se compone de un único método proyecto, se pueden omitir los parámetros en la inicialización del objeto fórmula. Simplemente se pueden pasar la fórmula se llama. Por ejemplo:

```
var $f : 4D.Function  
  
$f:=Formula(myMethod)  
//Writing Formula(myMethod($1;$2)) no es necesario  
$text:=$f.call(Null;"Hello";"World") //devuelve "Hello World"  
$text:=$f.call() //devuelve "How are you?"  
  
//myMethod  
#DECLARE ($param1 : Text; $param2 : Text)->$return : Text  
If(Count parameters=2)  
    $return:=$param1+" "+$param2  
Else  
    $return:="How are you?"  
End if
```

Los parámetros se reciben en el método, en el orden en que se especifican en la llamada.

## Sobre los objetos 4D.Function

Un objeto `4D.Function` contiene un trozo de código que puede ser ejecutado desde un objeto, ya sea utilizando el operador `()`, o utilizando las funciones `apply()` y `call()`. 4D propone tres tipos de objetos Function:

- las funciones nativas, es decir, las funciones integradas de varias clases 4D tales como `collection.sort()` o `file.copyTo()`.
- las funciones usuario, creadas en las `clases` usuario utilizando la `palabra clave Function`.
- las funciones de fórmula, es decir, las funciones que pueden ejecutar cualquier fórmula 4D.

## Resumen

```
.apply() : any  
.apply( thisObj : Object { ; formulaParams : Collection } ) : any  
ejecuta el objeto formula al que se aplica y devuelve el valor resultante<!-- END REF --. El objeto fórmula  
puede ser creado con los comandos Formula o
```

`Formula from string .`

En el parámetro `thisObj`, puede pasar una referencia al objeto que se utilizará como `This` dentro de la fórmula.

También puede pasar una colección que se utilizará como parámetros `$1...$n` en la fórmula utilizando el parámetro opcional `formulaParams`.

Tenga en cuenta que `.apply()` es similar a `.call()` excepto que los parámetros se pasan como una colección. Esto puede ser útil para pasar los resultados calculados.

## Ejemplo 1

```
var $f : 4D.Function  
$f:=Formula($1+$2+$3)  
  
$c:=New collection(10;20;30)  
$result:=$f.apply(Null;$c) // devuelve 60
```

## Ejemplo 2

```

var $calc : 4D.Function
var $feta; $robot : Object
$robot:=New object("name";"Robot";"price";543;"quantity";2)
$feta:=New object("name";"Feta";"price";12.5;"quantity";5)

$calc:=Formula(This.total:=This.price*This.quantity)

$calc.apply($feta) // $feta={name:Feta,price:12.5,quantity:5,total:62.5}
$calc.apply($robot) // $robot={name:Robot,price:543,quantity:2,total:1086}

```

|| .call() : any  
.call( *thisObj* : Object { ; ...*params* : any } ) : any

ejecuta el objeto `formula` al que se aplica y devuelve el valor resultante<!-- END REF --. El objeto fórmula puede ser creado con los comandos `Formula` o

`Formula from string`.

En el parámetro *thisObj*, puede pasar una referencia al objeto que se utilizará como `This` dentro de la fórmula.

También puede pasar los valores a utilizar como parámetros `$1...$n` en la fórmula utilizando el parámetro opcional *params*).

Tenga en cuenta que `.call()` es similar a `.apply()` excepto que los parámetros se pasan directamente.

## Ejemplo 1

```

var $f : 4D.Function
$f:=Formula(Uppercase($1))
$result:=$f.call(Null;"hello") // devuelve "HELLO"

```

## Ejemplo 2

```

$o:=New object("value";50)
$f:=Formula(This.value*2)
$result:=$f.call($o) // devuelve 100

```

| .source : Text

contiene la expresión fuente de la `fórmula` como texto |

## Formula

► Histórico

`Formula ( formulaExp : Expression ) : 4D.Function`

Parámetros	Tipo		Descripción
formulaExp	Expresión	->	Fórmula a devolver como objeto
Resultado	4D.Function	<-	Función nativa que encapsula la fórmula

### Descripción

El comando `Formula` crea un objeto `4D Function` basado en la expresión *formulaExp*. *formulaExp* puede ser tan

simple como un valor único o complejo, como un método proyecto con parámetros.

Tener una fórmula como objeto permite pasarla como parámetro (atributo calculado) a los comandos o a los métodos o ejecutarla desde varios componentes sin necesidad de declararla como "compartida por los componentes y la base de datos local". Cuando se llama, el objeto fórmula se evalúa en el contexto de la base de datos o del componente que lo creó.

La fórmula devuelta puede ser llamada con:

- los métodos `.call()` o `.apply()`, o
- object notation syntax (see [formula object](#)).

```
var $f : 4D.Function
$f:=Formula(1+2)
$o:=New object("myFormula";$f)

//tres formas diferentes de llamar a la fórmula
$f.call($o) //devuelve 3
$f.apply($o) //devuelve 3
$o.myFormula() //devuelve 3
```

Puede pasar los [parámetros](#) a `Formula`, como se muestra en [example-4">ejemplo 4](#) abajo.

Se puede especificar el objeto sobre el que se ejecuta la fórmula, como se ve en el [ejemplo 5](#). Se puede acceder a las propiedades del objeto mediante el comando `This`.

Si `formulaExp` utiliza variables locales, sus valores se copian y almacenan en el objeto fórmula devuelto durante su creación. Cuando se ejecuta, la fórmula utiliza estos valores copiados en lugar del valor actual de las variables locales. Tenga en cuenta que no se soporta el uso de arrays como variables locales.

El objeto creado por `Formula` puede guardarse, por ejemplo, en un campo de la base de datos o en un documento blob.

## Ejemplo 1

Una fórmula simple:

```
var $f : 4D.Function
$f:=Formula(1+2)

var $o : Object
$o:=New object("f";$f)

$result:=$o.f() // devuelve 3
```

## Ejemplo 2

Una fórmula utilizando variables locales:

```
$value:=10
$o:=New object("f";Formula($value))
$value:=20

$result:=$o.f() // devuelve 10
```

## Ejemplo 3

Una fórmula sencilla que utiliza parámetros:

```
$o:=New object("f";Formula($1+$2))
$result:=$o.f(10;20) //devuelve 30
```

## Ejemplo 4

Una fórmula utilizando un método proyecto con parámetros:

```
$o:=New object("f";Formula(myMethod))
$result:=$o.f("param1";"param2") // equivalente a $result:=myMethod("param1";"param2")
```

## Ejemplo 5

Utilizando `This`:

```
$o:=New object("fullName";Formula(This.firstName+" "+This.lastName))
$o.firstName:="John"
$o.lastName:="Smith"
$result:=$o.fullName() //devuelve "John Smith"
```

## Ejemplo 6

Llamar a una fórmula utilizando la notación de objetos:

```
var $feta; $robot : Object
var $calc : 4D.Function
$robot:=New object("name";"Robot";"price";543;"quantity";2)
$feta:=New object("name";"Feta";"price";12.5;"quantity";5)

$calc:=Formula(This.total:=This.price*This.quantity)

//define la fórmula de las propiedades del objeto
$feta.calc:=$calc
$robot.calc:=$calc

//llama la fórmula
$feta.calc() // $feta={name:Feta,price:12.5,quantity:5,total:62.5,calc:"[object Formula]"}
$robot.calc() // $robot={name:Robot,price:543,quantity:2,total:1086,calc:"[object Formula]"}
```

## Formula from string

► Histórico

`Formula from string( formulaString : Text ) : 4D.Function`

Parámetros	Tipo		Descripción
formulaString	Texto	->	Fórmula texto a devolver como objeto
Resultado	4D.Function	<-	Objeto nativo que encapsula la fórmula

### Descripción

The `Formula from string` command creates a 4D.Function object based upon the `formulaString`. `formulaString` puede ser tan simple como un valor único o complejo, como un método proyecto con parámetros.

Este comando es similar a `Formula`, excepto que maneja una fórmula basada en texto. En la mayoría de los casos, se recomienda utilizar el comando `Formula`. `Formula from string` sólo debe utilizarse cuando la fórmula original fue expresada como texto (por ejemplo, almacenada externamente en un archivo JSON). En este contexto, el uso de la

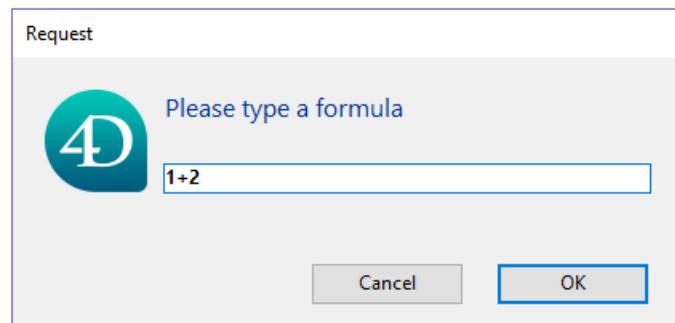
sintaxis con tokens es muy aconsejable.

Dado que no se puede acceder al contenido de las variables locales por su nombre en el modo compilado, no se pueden utilizar en *formulaString*. Un intento de acceder a una variable local con `Formula from string` dará lugar a un error (-10737).

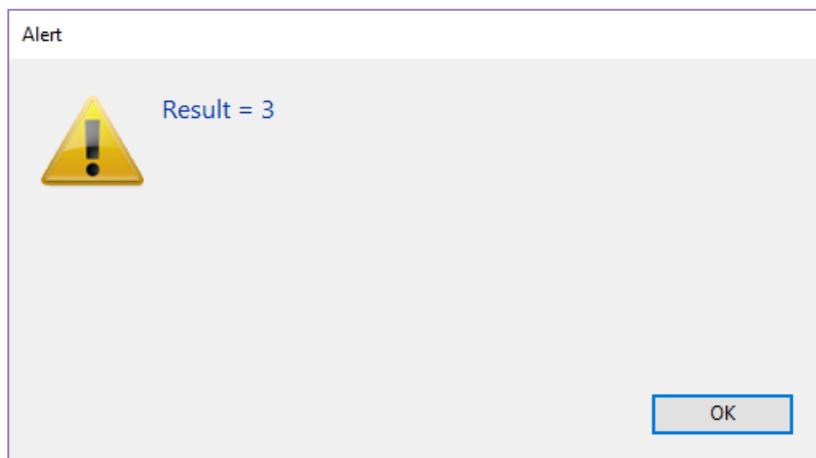
## Ejemplo

El siguiente código creará un diálogo que acepta una fórmula en formato texto:

```
var $textFormula : Text
var $f : 4D.Function
$textFormula:=Request("Por favor, escriba una fórmula")
If(ok=1)
  $f:=Formula from string($textFormula)
  ALERT("Result = "+String($f.call()))
End if
```



...y ejecuta la fórmula:



## .apply()

► Histórico

`.apply() : any`

`.apply( thisObj : Object { ; formulaParams : Collection } ) : any`

Parámetros	Tipo	Descripción
thisObj	Objeto	-> Objeto a devolver por el comando This en la fórmula
formulaParams	Collection	-> Colección de valores que se pasan como \$1...\$n cuando se ejecuta la fórmula
Resultado	any	<- Valor obtenido de la ejecución de la fórmula

## Descripción

La función `.apply()` ejecuta el objeto `formula` al que se aplica y devuelve el valor resultante<!-- END REF --. El objeto fórmula puede ser creado con los comandos `Formula` o

`Formula from string`.

En el parámetro `thisObj`, puede pasar una referencia al objeto que se utilizará como `This` dentro de la fórmula.

También puede pasar una colección que se utilizará como parámetros `$1...$n` en la fórmula utilizando el parámetro opcional `formulaParams`.

Tenga en cuenta que `.apply()` es similar a `.call()` excepto que los parámetros se pasan como una colección. Esto puede ser útil para pasar los resultados calculados.

### Ejemplo 1

```
var $f : 4D.Function  
$f:=Formula($1+$2+$3)  
  
$c:=New collection(10;20;30)  
$result:=$f.apply(NULL;$c) // devuelve 60
```

### Ejemplo 2

```
var $calc : 4D.Function  
var $feta; $robot : Object  
$robot:=New object("name";"Robot";"price";543;"quantity";2)  
$feta:=New object("name";"Feta";"price";12.5;"quantity";5)  
  
$calc:=Formula(This.total:=This.price*This.quantity)  
  
$calc.apply($feta) // $feta={name:Feta,price:12.5,quantity:5,total:62.5}  
$calc.apply($robot) // $robot={name:Robot,price:543,quantity:2,total:1086}
```

## .call()

► Histórico

`.call() : any`

`.call( thisObj : Object { ; ...params : any } ) : any`

Parámetros	Tipo		Descripción
thisObj	Objeto	->	Objeto a devolver por el comando This en la fórmula
params	any	->	Valor(es) que se pasa(n) como <code>\$1...\$n</code> cuando se ejecuta la fórmula
Resultado	any	<-	Valor obtenido de la ejecución de la fórmula

## Descripción

La función `.call()` ejecuta el objeto `formula` al que se aplica y devuelve el valor resultante<!-- END REF --. El objeto fórmula puede ser creado con los comandos `Formula` o

`Formula from string`.

En el parámetro `thisObj`, puede pasar una referencia al objeto que se utilizará como `This` dentro de la fórmula.

También puede pasar los valores a utilizar como parámetros `$1...$n` en la fórmula utilizando el parámetro opcional `params`.

Tenga en cuenta que `.call()` es similar a `.apply()` excepto que los parámetros se pasan directamente.

## Ejemplo 1

```
var $f : 4D.Function  
$f:=Formula(Uppercase($1))  
$result:=$f.call(Null;"hello") // devuelve "HELLO"
```

## Ejemplo 2

```
$o:=New object("value";50)  
$f:=Formula(This.value*2)  
$result:=$f.call($o) // devuelve 100
```

## .SOURCE

► Histórico

`.source` : Text

## Descripción

La propiedad `.source` contiene la expresión fuente de la `fórmula` como texto.

Esta propiedad es de sólo lectura.

## Ejemplo

```
var $of : 4D.Function  
var $tf : Text  
$of:=Formula(String(Current time;HH MM AM PM))  
$tf:=$of.source //"String(Current time;HH MM AM PM)"
```

# IMAPTransporter

La clase `IMAPTransporter` permite recuperar mensajes de un servidor de mensajería IMAP.

## Objeto IMAP Transporter

Los objetos IMAP Transporter se instancian con el comando [IMAP New transporter](#). Ofrecen las siguientes propiedades y funciones:

`.acceptUnsecureConnection: Boolean`

True si se permite a 4D establecer una conexión no cifrada

`.addFlags( msgIDs : Collection ; keywords : Object ) : Object`

`.addFlags( msgIDs : Text ; keywords : Object ) : Object`

`.addFlags( msgIDs : Longint ; keywords : Object ) : Object`

añade banderas a los `msgIDs` para las `keywords`

`.append( mailObj : Object ; destinationBox : Text ; options : Object ) : Object`

añade el objeto `mailObj` a la caja `destinationBox`

`.authenticationMode: Text`

el modo de autenticación utilizado para abrir la sesión en el servidor de correo

`.checkConnection() : Object`

comprueba la conexión utilizando la información almacenada en el objeto transporter

`.checkConnectionDelay : Integer`

duración máxima (en segundos) permitida antes de verificar la conexión al servidor

`.connectionTimeOut : Integer`

el tiempo máximo de espera (en segundos) permitido para establecer una conexión con el servidor

`.copy( msgsIDs : Collection ; destinationBox : Text ) : Object`

`.copy( allMsgs : Integer ; destinationBox : Text ) : Object`

copia los mensajes definidos en `msgsIDs` o `allMsgs` en la `destinationBox` en el servidor IMAP

`.createBox( name : Text ) : Object`

crea un buzón con el `name` pasado

`.delete( msgsIDs : Collection ) : Object`

`.delete( allMsgs : Integer ) : Object`

asocia el marcador "deleted" a los mensajes designados por `msgsIDs` o `allMsgs`

`.deleteBox( name : Text ) : Object`

elimina definitivamente el buzón llamado `name` en el servidor IMAP

`.expunge() : Object`

elimina todos los mensajes con el marcador "deleted" del servidor de correo IMAP.

`.getBoxInfo( { name : Text } ) : Object`

devuelve un objeto `boxInfo` correspondiente al buzón de recepción actual o al buzón llamado `name`

`.getBoxList( { parameters : Object } ) : Collection`

devuelve una colección de bandejas de entrada que describe todas las bandejas de entrada disponibles

`.getDelimiter() : Text`

devuelve el carácter utilizado para delimitar los niveles de jerarquía en el nombre del buzón

`.getMail( msgNumber: Integer { ; options : Object } ) : Object`

`.getMail( msgID: Text { ; options : Object } ) : Object`

devuelve el objeto `Email` correspondiente al `msgNumber` o `msgID` en el buzón designado por `IMAP_transporter`

`.getMails( ids : Collection { ; options : Object } ) : Object`

`.getMails( startMsg : Integer ; endMsg : Integer { ; options : Object } ) : Object`

devuelve un objeto que contiene una colección de objetos `Email`

`.getMIMEAsBlob( msgNumber : Integer { ; updateSeen : Boolean } ) : Blob`

`.getMIMEAsBlob( msgID : Text { ; updateSeen : Boolean } ) : Blob`

devuelve un BLOB con el contenido MIME del mensaje correspondiente al `msgNumber` o `msgID` en el buzón designado por el `IMAP_transporter`

`.host : Text`

el nombre o la dirección IP del servidor local

`.logFile : Text`

la ruta del archivo de registro extendido definido (si existe) para la conexión de correo

`.move( msgsIDs : Collection ; destinationBox : Text ) : Object`

`.move( allMsgs : Integer ; destinationBox : Text ) : Object`

move los mensajes definidos en `msgsIDs` o `allMsgs` a la `destinationBox`en el servidor IMAP

`.numToID( startMsg : Integer ; endMsg : Integer ) : Collection`

convierte los números de secuencia en IDs únicos IMAP para los mensajes en el rango secuencial designado por `startMsg` y `endMsg`

`.removeFlags( msgIDs : Collection ; keywords : Object ) : Object`

`.removeFlags( msgIDs : Text ; keywords : Object ) : Object`

`.removeFlags( msgIDs : Longint ; keywords : Object ) : Object`

elimina las banderas de los `msgIDs` para las `keywords` definidas

`.renameBox( currentName : Text ; newName : Text ) : Object`

cambia el nombre de un buzón en el servidor IMAP

`.port : Integer`

el número de puerto utilizado para las transacciones de correo

`.searchMails( searchCriteria : Text ) : Collection`

busca los mensajes que coincidan con el `searchCriteria` en el buzón actual

`.selectBox( name : Text { ; state : Integer } ) : Object`

selecciona el buzón `name` como el buzón actual

`.subscribe( name : Text ) : Object`

permite añadir o eliminar el buzón especificado del conjunto de buzones "suscritos" en el servidor IMAP

`.unsubscribe( name : Text ) : Object`

elimina un buzón de un conjunto de buzones suscritos

`.user : Text`

el nombre de usuario utilizado para la autenticación en el servidor de correo

# IMAP New transporter

► Histórico

IMAP New transporter( *server* : Object ) : 4D.IMAPTransporter

Parámetros	Tipo		Descripción
<i>server</i>	Objeto	->	Información del servidor de correo
Resultado	4D.IMAPTransporter	<-	<a href="#">Objeto IMAP Transporter</a>

## Descripción

El comando `IMAP New transporter` configura una nueva conexión IMAP según el parámetro *server* y devuelve un nuevo objeto *transporter*. El objeto transportador devuelto se utilizará normalmente para recibir correos electrónicos.

En el parámetro *server*, pase un objeto que contenga las siguientes propiedades:

<i>server</i>		Valor por defecto (si se omite)
<a href="#">.acceptUnsecureConnection: Boolean</a> True si se permite a 4D establecer una conexión no cifrada		False
<a href="#">.accessTokenOAuth2: Text</a> <a href="#">.accessTokenOAuth2: Object</a> Cadena de texto u objeto token que representan las credenciales de autorización OAuth 2. Sólo se utiliza con <code>OAUTH2 authenticationMode</code> . Si se utiliza <code>accessTokenOAuth2</code> pero se omite <code>authenticationMode</code> , se utiliza el protocolo OAuth 2 (si el servidor lo permite). No se devuelve en el objeto <i>IMAP transporter</i> .		ninguno
<a href="#">.authenticationMode: Text</a> el modo de autenticación utilizado para abrir la sesión en el servidor de correo		se utiliza el modo de autenticación más seguro soportado por el servidor
<a href="#">.checkConnectionDelay : Integer</a> duración máxima (en segundos) permitida antes de verificar la conexión al servidor	300	
<a href="#">.connectionTimeOut : Integer</a> el tiempo máximo de espera (en segundos) permitido para establecer una conexión con el servidor	30	
<a href="#">.host : Text</a> el nombre o la dirección IP del servidor local		<i>obligatorio</i>
<a href="#">.logFile : Text</a> la ruta del archivo de registro extendido definido (si existe) para la conexión de correo		ninguno
<a href="#">.password : Text</a> Contraseña del usuario para la autenticación en el servidor. No se devuelve en el objeto <i>IMAP transporter</i> .		ninguno
<a href="#">.port : Integer</a> el número de puerto utilizado para las transacciones de correo	993	
<a href="#">.user : Text</a> el nombre de usuario utilizado para la autenticación en el servidor de correo		ninguno

Atención: asegúrese de que el tiempo de espera definido sea menor que el tiempo de espera del servidor, de lo contrario el tiempo de espera del cliente será inútil.

## Resultado

La función devuelve un [IMAP transporter](#). Todas las propiedades devueltas son de sólo lectura.

La conexión IMAP se cierra automáticamente cuando se destruye el objeto transportador.

## Ejemplo

```
$server:=New object
$server.host:="imap.gmail.com" //Obligatorio
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"
$server.logFile:="LogTest.txt" //log a guardar en la carpeta Logs

var $transporter : 4D.IMAPTransporter
$transporter:=IMAP New transporter($server)

$status:=$transporter.checkConnection()
If(Not($status.success))
  ALERT("An error occurred: "+$status.statusText)
End if
```

## 4D.IMAPTransporter.new()

4D.IMAPTransporter.new( server : Object ) : 4D.IMAPTransporter

Parámetros	Tipo		Descripción
server	Objeto	->	Información del servidor de correo
Resultado	4D.IMAPTransporter	<-	<a href="#">Objeto IMAP Transporter</a>

## Descripción

La función `4D.IMAPTransporter.new()` crea y devuelve un nuevo objeto del tipo `4D.IMAPTransporter`. Es idéntico al comando [IMAP New transporter](#) (acceso directo).

## .acceptUnsecureConnection

► Histórico

.acceptUnsecureConnection: Boolean

## Descripción

La propiedad `.acceptUnsecureConnection` contiene True si se permite a 4D establecer una conexión no cifrada cuando la conexión cifrada no es posible.

Contiene False si no se permiten las conexiones no cifradas, en cuyo caso se devuelve un error cuando no es posible la conexión cifrada.

Los puertos seguros disponibles son:

- SMTP
  - 465: SMTPS

- 587 o 25: SMTP con actualización STARTTLS si lo soporta el servidor.
- IMAP
  - 143: Puerto IMAP no encriptado
  - 993: IMAP con actualización STARTTLS si lo soporta el servidor
- POP3
  - 110: Puerto POP3 no encriptado
  - 995: POP3 con actualización STARTTLS si lo soporta el servidor.

## .addFlags()

► Histórico

`.addFlags( msgIDs : Collection ; keywords : Object ) : Object`

`.addFlags( msgIDs : Text ; keywords : Object ) : Object`

`.addFlags( msgIDs : Longint ; keywords : Object ) : Object`

Parámetros	Tipo		Descripción
msgIDs	Collection	->	Colección de cadenas: identificadores únicos de mensajes (texto) Texto: ID único de un mensaje Longint (IMAP all): todos los mensajes del buzón seleccionado
keywords	Objeto	->	Banderas de palabras claves a añadir
Resultado	Objeto	<-	Estado de la operación addFlags

### Descripción

La función `.addFlags()` añade banderas a los `msgIDs` para las `keywords`.

En el parámetro `msgIDs`, puede pasar:

- una *collection* contiene los identificadores únicos de mensajes específicos, o
- el ID único (*texte*) de un solo mensaje o
- la siguiente constante (*longint*) para todos los mensajes del buzón seleccionado:

Constante	Valor	Comentario
IMAP all	1	Seleccionar todos los mensajes del buzón seleccionado

El parámetro `keywords` le permite pasar un objeto con valores de palabras clave para las banderas específicas que se añadirán a `msgIDs`. Puede pasar cualquiera de las siguientes palabras claves:

Parámetros	Tipo	Descripción
\$draft	Booleano	True para añadir el marcador "draft" al mensaje
\$seen	Booleano	True para añadir el marcador "seen" al mensaje
\$flagged	Booleano	True para añadir el marcador "flagged" al mensaje
\$answered	Booleano	True para añadir el marcador "answered" al mensaje
\$deleted	Booleano	True para añadir el marcador "deleted" al mensaje

- Los valores falsos se ignoran.
- La interpretación de los indicadores de palabras claves puede variar según el cliente de correo.

### Objeto devuelto

La función devuelve un objeto que describe el estado IMAP:

Propiedad		Tipo	Descripción
success		Booleano	True si la operación tiene éxito, False en caso contrario
statusText		Texto	Mensaje de estado devuelto por el servidor IMAP, o último error devuelto en la pila de errores de 4D
errors		Collection	Pila de errores 4D (no se devuelve si se recibe una respuesta del servidor IMAP)
	[].errcode	Número	Código de error 4D
	[].message	Texto	Descripción del error 4D
	[].componentSignature	Texto	Firma del componente interno que ha devuelto el error

## Ejemplo

```

var $options;$transporter;$boxInfo;$status : Object

$options:=New object
$options.host:="imap.gmail.com"
$options.port:=993
$options.user:="4d@gmail.com"
$options.password:="xxxxx"

// Crear transportador
$transporter:=IMAP New transporter($options)

// Seleccionar buzón de correo
$boxInfo:=$transporter.selectBox("INBOX")

// Marcar todos los mensajes de INBOX como leídos/vistos
$flags:=New object
$flags["$seen"]:=True
$status:=$transporter.addFlags(IMAP all;$flags)

```

## .append()

### ► Histórico

.append( *mailObj* : Object ; *destinationBox* : Text ; *options* : Object ) : Object

Parámetros	Tipo		Descripción
mailObj	Objeto	->	Objeto Email
destinationBox	Texto	->	Buzón para recibir el objeto Email
options	Objeto	->	Objeto que contiene información del charset
Resultado	Objeto	<-	Estado de la operación

### Descripción

La función `.append()` añade el objeto `mailObj` a la caja `destinationBox`.

En el parámetro `mailObj`, pase un objeto Email. En el parámetro `mailObj`, pase un objeto Email. La función `.append()` soporta los marcadores de palabras clave en el atributo `keywords` de los objetos email.

El parámetro opcional `destinationBox` permite pasar el nombre del buzón donde se añadirá el objeto `mailObj`. Si se omite, se utiliza el buzón actual.

En el parámetro opcional `options`, puede pasar un objeto para definir el charset y la codificación para partes específicas del correo electrónico. Propiedades disponibles:

Propiedad	Tipo	Descripción
headerCharset	Texto	Charset y codificación utilizados para las siguientes partes del correo electrónico: asunto, nombres de archivos adjuntos y atributo(s) del nombre del correo electrónico. Valores posibles: ver la tabla de charsets posibles a continuación
bodyCharset	Texto	Charset y codificación utilizados para el contenido html y el texto del cuerpo del correo electrónico. Valores posibles: ver la tabla de charsets posibles a continuación

Charsets posibles:

Constante	Valor	Comentario
mail mode ISO2022JP	US-ASCII_ISO-2022-JP_UTF8_QP	<ul style="list-style-type: none"> <li>headerCharset: US-ASCII si es posible, japonés (ISO-2022-JP) &amp; Quoted-printable si es posible, de lo contrario UTF-8 &amp; Quoted-printable</li> <li>bodyCharset: US-ASCII si es posible, japonés (ISO-2022-JP) y 7 bits si es posible, de lo contrario UTF-8 &amp; Quoted-printable</li> </ul>
mail mode ISO88591	ISO-8859-1	<ul style="list-style-type: none"> <li>headerCharset: ISO-8859-1 &amp; Quoted-printable</li> <li>bodyCharset: ISO-8859-1 &amp; 8-bit</li> </ul>
mail mode UTF8	US-ASCII_UTF8_QP	headerCharset & bodyCharset: US-ASCII si es posible, de lo contrario UTF-8 & Quoted-printable (valor por defecto)
mail mode UTF8 in base64	US-ASCII_UTF8_B64	headerCharset & bodyCharset: US-ASCII si es posible, de lo contrario UTF-8 & base64

Objeto devuelto

La función devuelve un objeto que describe el estado IMAP:

Propiedad		Tipo	Descripción
success		Booleano	True si la operación tiene éxito, False en caso contrario
statusText		Texto	Mensaje de estado devuelto por el servidor IMAP, o último error devuelto en la pila de errores de 4D
errors		Collection	Pila de errores 4D (no se devuelve si se recibe una respuesta del servidor IMAP)
	[].errcode	Número	Código de error 4D
	[].message	Texto	Descripción del error 4D
	[].componentSignature	Texto	Firma del componente interno que ha devuelto el error

Ejemplo

Para guardar un correo electrónico en el buzón de borradores:

```

var $settings; $status; $msg; $imap: Object

$settings:=New object("host"; "domain.com"; "user"; "xxxx"; "password"; "xxxx"; "port"; 993)

$imap:=IMAP New transporter($settings)

$msg:=New object
$msg.from:="xxxx@domain.com"
$msg.subject:="Lorem Ipsum"
$msg.textBody:="Lorem ipsum dolor sit amet, consectetur adipiscing elit."
$msg.keywords:=New object
$msg.keywords["$seen"]:=True//marcar el mensaje como leido
$msg.keywords["$draft"]:=True//marcar el mensaje como borrador

$status:=$imap.append($msg; "Drafts")

```

## .authenticationMode

► Histórico

.authenticationMode: Text

### Descripción

La propiedad `.authenticationMode` contiene el modo de autenticación utilizado para abrir la sesión en el servidor de correo.

Por defecto, se utiliza el modo más seguro soportado por el servidor.

Los valores posibles son:

Valor	Constantes	Comentario
CRAM-MD5	IMAP authentication CRAM MD5	Autenticación utilizando el protocolo CRAM-MD5
LOGIN	IMAP authentication login	Autenticación utilizando el protocolo LOGIN
OAuth2	IMAP authentication OAuth2	Autenticación utilizando el protocolo OAuth2
PLAIN	IMAP authentication plain	Autenticación utilizando el protocolo PLAIN

## .checkConnection()

► Histórico

.checkConnection() : Object

Parámetros	Tipo		Descripción
Resultado	Objeto	<-	Estado de la conexión del objeto transportador

### Descripción

La función `.checkConnection()` comprueba la conexión utilizando la información almacenada en el objeto transporter, recrea la conexión si es necesario y devuelve el estado. Esta función permite verificar que los valores proporcionados por el usuario son válidos y coherentes.

### Objeto devuelto

La función envía una solicitud al servidor de correo y devuelve un objeto que describe el estado del correo. Este objeto puede contener las siguientes propiedades:

Propiedad		Tipo	Descripción
success		booleano	True si la verificación es exitosa, False en caso contrario
status		number	(sólo SMTP) Código de estado devuelto por el servidor de correo (0 en caso de un problema no relacionado con el procesamiento del correo)
statusText		texto	Mensaje de estado devuelto por el servidor de correo, o último error devuelto en la pila de errores de 4D
errors		colección	Pila de errores 4D (no se devuelve si se recibe una respuesta del servidor de correo)
[ ].errCode		number	Código de error 4D
[ ].message		texto	Descripción del error 4D
[ ].componentSignature		texto	Firma del componente interno que ha devuelto el error

## .checkConnectionDelay

► Histórico

.checkConnectionDelay : Integer

### Descripción

La propiedad `.checkConnectionDelay` contiene la duración máxima (en segundos) permitida antes de verificar la conexión al servidor. Si se supera este tiempo entre dos llamadas al método, se comprobará la conexión con el servidor. Por defecto, si la propiedad no se ha definido en el objeto `server`, el valor es de 300.

Atención: asegúrese de que el tiempo de espera definido sea menor que el tiempo de espera del servidor, de lo contrario el tiempo de espera del cliente será inútil.

## .connectionTimeOut

► Histórico

.connectionTimeOut : Integer

### Descripción

La propiedad `.connectionTimeOut` contiene el tiempo máximo de espera (en segundos) permitido para establecer una conexión con el servidor. Por defecto, si la propiedad no se ha definido en el objeto servidor (utilizado para crear el objeto transportador con `SMTP New transporter`, `POP3 New transporter`, o `IMAP New transporter`), el valor es 30.

## .copy()

► Histórico

.copy( `msgsIDs` : Collection ; `destinationBox` : Text ) : Object

.copy( `allMsgs` : Integer ; `destinationBox` : Text ) : Object

Parámetros	Tipo		Descripción
<code>msgsIDs</code>	Collection	->	Colección de identificadores únicos de mensajes (cadenas)
<code>allMsgs</code>	Integer	->	IMAP <code>all</code> : todos los mensajes del buzón seleccionado
<code>destinationBox</code>	Texto	->	Buzón para recibir mensajes copiados
Resultado	Objeto	<-	Estado de la operación de copia

## Descripción

La función `.copy()` copia los mensajes definidos en `msgsIDs` o `allMsgs` en la `destinationBox` en el servidor IMAP.

Puede pasar:

- en el parámetro `msgsIDs`, una colección contiene los IDs únicos de los mensajes específicos a copiar, o
- en el parámetro `allMsgs`, la constante `IMAP all` (entero) para copiar todos los mensajes del buzón seleccionado.

El parámetro `destinationBox` permite pasar un valor texto con el nombre del buzón donde se colocarán las copias de los mensajes.

Objeto devuelto

La función devuelve un objeto que describe el estado IMAP:

Propiedad		Tipo	Descripción
<code>success</code>		Booleano	True si la operación tiene éxito, False en caso contrario
<code>statusText</code>		Texto	Mensaje de estado devuelto por el servidor IMAP, o último error devuelto en la pila de errores de 4D
<code>errors</code>		Collection	Pila de errores 4D (no se devuelve si se recibe una respuesta del servidor IMAP)
	<code>[] .errcode</code>	Número	Código de error 4D
	<code>[] .message</code>	Texto	Descripción del error 4D
	<code>[] .componentSignature</code>	Texto	Firma del componente interno que ha devuelto el error

## Ejemplo 1

Para copiar una selección de mensajes:

```
var $server;$boxInfo;$status : Object
var $mailIds : Collection
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com" //Obligatorio
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXXX"

$transporter:=IMAP New transporter($server)

//seleccionar buzón
$boxInfo:=$transporter.selectBox("inbox")

//obtener la colección de IDs únicos de los mensajes
$mailIds:=$transporter.searchMails("subject \\"4D new feature:\\\"")

// copiar los mensajes encontrados en el buzón "documents"
$status:=$transporter.copy($mailIds;"documents")
```

## Ejemplo 2

Para copiar todos los mensajes del buzón actual:

```

var $server;$boxInfo;$status : Object
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com" //Obligatorio
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

$transporter:=IMAP New transporter($server)

//seleccionar buzón

$boxInfo:=$transporter.selectBox("inbox")

// copiar los mensajes encontrados en el buzón "documents"
$status:=$transporter.copy(IMAP all;"documents")

```

## .createBox()

► Histórico

.createBox( name : Text ) : Object

Parámetros	Tipo		Descripción
name	Texto	->	Nombre del nuevo buzón
Resultado	Objeto	<-	Estado de la operación de creación del buzón

### Descripción

La función `.createBox()` crea un buzón con el `name` pasado. Si el carácter separador de jerarquía del servidor IMAP aparece en otra parte del nombre del buzón, el servidor IMAP creará todos los nombre padre necesarios para crear el buzón dado.

En otras palabras, un intento de crear "Projects/IMAP/Doc" en un servidor en el que "/" es el carácter separador de jerarquía creará:

- Sólo el buzón "Doc" si "Projects" & "IMAP" ya existen.
- Los buzones "IMAP" & "Doc" si sólo existe "Projects".
- Los buzones "Projects" & "IMAP" & "Doc", si no existen.

En el parámetro `name`, pasa el nombre del nuevo buzón.

### Objeto devuelto

La función devuelve un objeto que describe el estado IMAP:

Propiedad		Tipo	Descripción
success		Booleano	True si la operación tiene éxito, False en caso contrario
statusText		Texto	Mensaje de estado devuelto por el servidor IMAP, o último error devuelto en la pila de errores de 4D
errors		Collection	Pila de errores 4D (no se devuelve si se recibe una respuesta del servidor IMAP)
	[].errcode	Número	Código de error 4D
	[].message	Texto	Descripción del error 4D
	[].componentSignature	Texto	Firma del componente interno que ha devuelto el error

## Ejemplo

Para crear un nuevo buzón "Invoices":

```
var $pw : text
var $options; $transporter; $status : object

$options:=New object

$pw:=Request("Please enter your password:")
If(OK=1)
$options.host:="imap.gmail.com"
$options.user:="test@gmail.com"
$options.password:=$pw

$transporter:=IMAP New transporter($options)

$status:=$transporter.createBox("Invoices")

If ($status.success)
ALERT("Mailbox creation successful!")
Else
ALERT("Error: "+$status.statusText)
End if
End if
```

## .delete()

► Histórico

.delete( *msgsIDs* : Collection ) : Object  
.delete( *allMsgs* : Integer ) : Object

Parámetros	Tipo		Descripción
<i>msgsIDs</i>	Collection	->	Colección de identificadores únicos de mensajes (cadenas)
<i>allMsgs</i>	Integer	->	IMAP all : todos los mensajes del buzón seleccionado
Resultado	Objeto	<-	Estado de la operación de eliminación

### Descripción

La función `.delete()` asocia el marcador "deleted" a los mensajes designados por `msgsIDs` o `allMsgs`.

Puede pasar:

- en el parámetro `msgsIDs`, una colección contiene los IDs únicos de los mensajes específicos a eliminar, o
- en el parámetro `allMsgs`, la constante `IMAP all` (entero) para eliminar todos los mensajes del buzón seleccionado.

La ejecución de esta función no elimina realmente los mensajes. Los mensajes con la bandera "delete" pueden seguir siendo encontrados por la función `.searchMails()`. Los mensajes marcados se eliminan del servidor IMAP con la función `.expunge()` o al seleccionar otro buzón o cuando se destruye el objeto `transporter` (creado con `IMAP New transporter`).

Objeto devuelto

La función devuelve un objeto que describe el estado IMAP:

Propiedad		Tipo	Descripción
success		Booleano	True si la operación tiene éxito, False en caso contrario
statusText		Texto	Mensaje de estado devuelto por el servidor IMAP, o último error devuelto en la pila de errores de 4D
errors		Collection	Pila de errores 4D (no se devuelve si se recibe una respuesta del servidor IMAP)
	[].errcode	Número	Código de error 4D
	[].message	Texto	Descripción del error 4D
	[].componentSignature	Texto	Firma del componente interno que ha devuelto el error

## Ejemplo 1

Para eliminar una selección de mensajes:

```

var $server;$boxInfo;$status : Object
var $mailIds : Collection
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com" //Obligatorio
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

$transporter:=IMAP New transporter($server)

//seleccionar buzón
$boxInfo:=$transporter.selectBox("Inbox")

//obtener la colección de IDs únicos de los mensajes
$mailIds:=$transporter.searchMails("subject \"Reports\"")

// Borrar los mensajes seleccionados
$status:=$transporter.delete($mailIds)

```

## Ejemplo 2

Para eliminar todos los mensajes del buzón actual:

```

var $server;$boxInfo;$status : Object
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com" //Obligatorio
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

$transporter:=IMAP New transporter($server)

//seleccionar buzón
$boxInfo:=$transporter.selectBox("Junk Email")

// Borrar los mensajes seleccionados en el buzón actual
$status:=$transporter.delete(IMAP all)

```

## .deleteBox()

► Histórico

.deleteBox( *name* : Text ) : Object

Parámetros	Tipo		Descripción
<i>name</i>	Texto	->	Nombre del buzón a eliminar

|Result|Object|<-|Estado de la operación de eliminación del buzón|

### Descripción

La función `.deleteBox()` elimina definitivamente el buzón llamado `name` en el servidor IMAP. Intentar eliminar un INBOX o un buzón que no existe generará un error.

En el parámetro `name`, pase el nombre del buzón a eliminar.

- La función no puede eliminar un buzón que tiene buzones hijos si el buzón padre tiene el atributo "X-Noselect".
- Todos los mensajes del buzón eliminado también se borrarán.
- La posibilidad de eliminar un buzón depende del servidor de correo.

### Objeto devuelto

La función devuelve un objeto que describe el estado IMAP:

Propiedad		Tipo	Descripción
success		Booleano	True si la operación tiene éxito, False en caso contrario
statusText		Texto	Mensaje de estado devuelto por el servidor IMAP, o último error devuelto en la pila de errores de 4D
errors		Collection	Pila de errores 4D (no se devuelve si se recibe una respuesta del servidor IMAP)
	[].errcode	Número	Código de error 4D
	[].message	Texto	Descripción del error 4D
	[].componentSignature	Texto	Firma del componente interno que ha devuelto el error

### Ejemplo

Para eliminar el buzón secundario "Nova Orion Industries" del interior del buzón "Bills":

```

var $pw; $name : text
var $options; $transporter; $status : object

$options:=New object

$pw:=Request("Please enter your password:")

If(OK=1) $options.host:="imap.gmail.com"
$options.user:="test@gmail.com"
$options.password:=$pw

$transporter:=IMAP New transporter($options)

// eliminar buzón
$name:="Bills"+$transporter.getDelimiter()+"Nova Orion Industries"
$status:=$transporter.deleteBox($name)

If ($status.success)
    ALERT("Mailbox deletion successful!")
Else
    ALERT("Error: "+$status.statusText)
End if
End if

```

## .expunge()

► Histórico

.expunge() : Object

Parámetros	Tipo		Descripción
Resultado	Objeto	<-	Estado de la operación expunge

### Descripción

La función `.expunge()` elimina todos los mensajes con el marcador "deleted" del servidor de correo IMAP. El marcador "deleted" puede establecerse con los métodos `.delete()` o `.addFlags()`.

### Objeto devuelto

La función devuelve un objeto que describe el estado IMAP:

Propiedad		Tipo	Descripción
success		Booleano	True si la operación tiene éxito, False en caso contrario
statusText		Texto	Mensaje de estado devuelto por el servidor IMAP, o último error devuelto en la pila de errores de 4D
errors		Collection	Pila de errores 4D (no se devuelve si se recibe una respuesta del servidor IMAP)
	[].errcode	Número	Código de error 4D
	[].message	Texto	Descripción del error 4D
	[].componentSignature	Texto	Firma del componente interno que ha devuelto el error

### Ejemplo

```

var $options;$transporter;$boxInfo;$status : Object
var $ids : Collection

$options:=New object
$options.host:="imap.gmail.com"
$options.port:=993
$options.user:="4d@gmail.com"
$options.password:="xxxxx"

// Crear transportador
$transporter:=IMAP New transporter($options)

// Seleccionar buzón
$boxInfo:=$transporter.selectBox("INBOX")

// Buscar y eliminar todos los mensajes vistos en INBOX
$ids:=$transporter.searchMails("SEEN")
$status:=$transporter.delete($ids)

// Purgar todos los mensajes marcados como borrados
$status:=$transporter.expunge()

```

## .getBoxInfo()

► Histórico

.getBoxInfo( { name : Text } ) : Object

Parámetros	Tipo		Descripción
name	Texto	->	Nombre del buzón
Resultado	Objeto	<-	objeto boxInfo

### Descripción

La función `.getBoxInfo()` devuelve un objeto `boxInfo` correspondiente al buzón de recepción actual o al buzón llamado `name`. Esta función devuelve la misma información que `.selectBox()` sin cambiar el buzón actual.

En el parámetro opcional `name`, pase el nombre del buzón a acceder. El nombre representa una jerarquía inequívoca de izquierda a derecha, con niveles separados por un carácter delimitador específico. El delimitador se puede recuperar con la función `.getDelimiter()`.

Si el buzón `name` no es seleccionable o no existe, la función genera un error y devuelve `null`.

### Objeto devuelto

El objeto `boxInfo` devuelto contiene las siguientes propiedades:

Propiedad	Tipo	Descripción
name	texto	Nombre del buzón
mailCount	number	Número de mensajes en el buzón
mailRecent	number	Número de mensajes con el marcador "reciente" (que indica los mensajes nuevos)

### Ejemplo

```

var $transporter : IMAPTransporter
$transporter:=IMAP New transporter($server)

$info:=$transporter.getBoxInfo("INBOX")
ALERT("INBOX contains "+String($info.mailRecent)+" recent emails.")

```

## .getBoxList()

► Histórico

`.getBoxList( { parameters : Object } ) : Collection`

Parámetros	Tipo		Descripción
parameters	Objeto	->	Objeto de parámetro
Resultado	Collection	<-	Colección de objetos mailbox

### Descripción

La función `.getBoxList()` devuelve una colección de bandejas de entrada que describe todas las bandejas de entrada disponibles. Esta función permite gestionar localmente la lista de mensajes localizados en el servidor de correo IMAP.

En el parámetro opcional `parameters`, pase un objeto que contenga valores para filtrar los buzones devueltos. Puede pasar:

Propiedad	Tipo	Descripción
isSubscribed	Booleano	<ul style="list-style-type: none"> <li>True para devolver sólo los buzones suscritos</li> <li>False para devolver todos los buzones disponibles</li> </ul>

### Resultado

Cada objeto de la colección devuelta contiene las siguientes propiedades:

Propiedad	Tipo	Descripción
[].name	texto	Nombre del buzón
[].selectable	booleano	Indica si los derechos de acceso permiten o no seleccionar el buzón: <ul style="list-style-type: none"> <li>true - el buzón puede ser seleccionado</li> <li>false - el buzón no puede ser seleccionado</li> </ul>
[].inferior	booleano	Indica si los derechos de acceso permiten o no crear una jerarquía inferior en el buzón: <ul style="list-style-type: none"> <li>true - se puede crear un nivel inferior</li> <li>false - no se puede crear un nivel inferior</li> </ul>
[].interesting	booleano	Indica si el buzón ha sido marcado como "interesante" por el servidor: <ul style="list-style-type: none"> <li>true - El buzón ha sido marcado como "interesante" por el servidor. Por ejemplo, puede contener mensajes nuevos.</li> <li>false - El buzón no ha sido marcado como "interesante" por el servidor.</li> </ul>

Si la cuenta no contiene buzones, se devuelve una colección vacía.

- Si no hay ninguna conexión abierta, `.getBoxList()` abrirá una conexión.
- Si la conexión no se ha utilizado desde el retraso de conexión designado (ver `IMAP New transporter`), se llama automáticamente a la función `.checkConnection()`.

## Ejemplo

```
var $transporter : 4D.IMAPTransporter
$transporter:=IMAP New transporter($server)

$boxList:=$transporter.getBoxList()

For each($box;$boxList)
  If($box.interesting)
    $split:=Split string($box.name;$transporter.getDelimiter())
    ALERT("New emails are available in the box: "+$split[$split.length-1])
  End if
End for each
```

## .getDelimiter()

► Histórico

.getDelimiter() : Text

Parámetros	Tipo		Descripción
Resultado	Texto	<-	Carácter delimitador de jerarquía

### Descripción

La función `.getDelimiter()` devuelve el carácter utilizado para delimitar los niveles de jerarquía en el nombre del buzón.

El delimitador es un carácter que puede utilizarse para:

- crear buzones de nivel inferior
- buscar más arriba o más abajo en la jerarquía del buzón

### Resultado

Carácter delimitador del nombre del buzón.

- Si no hay ninguna conexión abierta, `getDelimiter()` abrirá una conexión.
- Si la conexión no se ha utilizado desde el retraso de conexión designado, la función `.checkConnection()` se llama automáticamente.

## Ejemplo

```
var $transporter : 4D.IMAPTransporter
$transporter:=IMAP New transporter($server)

$boxList:=$transporter.getBoxList()

For each($box;$boxList)
  If($box.interesting)
    $split:=Split string($box.name;$transporter.getDelimiter())
    ALERT("New emails are available in the box: "+$split[$split.length-1])
  End if
End for each
```

## .getMail()

► Histórico

```
.getMail( msgNumber: Integer { ; options : Object } ) : Object  
.getMail( msgID: Text { ; options : Object } ) : Object
```

Parámetros	Tipo		Descripción
msgNumber	Integer	->	Número de secuencia del mensaje
msgID	Texto	->	ID único del mensaje
options	Objeto	->	Instrucciones sobre el manejo de mensajes
Resultado	Objeto	<-	<a href="#">Objeto Email</a>

## Descripción

La función `.getMail()` devuelve el objeto `Email` correspondiente al `msgNumber` o `msgID` en el buzón designado por `IMAP_transporter`. Esta función permite recuperar la información sobre el email.

En el primer parámetro, puede pasar:

- `msgNumber`, un valor `integer` indicando el número de secuencia del mensaje a recuperar o
- `msgID`, un valor `text` indicando el ID único del mensaje a recuperar.

El parámetro opcional `options` permite pasar un objeto que define las instrucciones adicionales para la gestión del mensaje. Las siguientes propiedades están disponibles:

Propiedad	Tipo	Descripción
updateSeen	booleano	Si True, el mensaje se marca como "visto" en el buzón. Si es False, el mensaje no se marca como "visto". Valor por defecto: True
withBody	booleano	Pase True para devolver el cuerpo del mensaje. Si es False, sólo se devuelve el encabezado del mensaje. Valor por defecto: True

- La función genera un error y devuelve Null si `msgID` designa un mensaje inexistente,
- Si no se selecciona ningún buzón con la función `.selectBox()`, se genera un error,
- Si no hay ninguna conexión abierta, `.getMail()` abrirá una conexión con el último buzón especificado por `.selectBox()`.

## Resultado

`.getMail()` devuelve un `objeto Email` con las propiedades IMAP adicionales siguientes: `id`, `receivedAt` y `size`.

## Ejemplo

Quiere obtener el mensaje con ID = 1:

```

var $server : Object
var $info; $mail; $boxInfo : Variant
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com" //Obligatorio
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

//crear transportador
$transporter:=IMAP New transporter($server)

//seleccionar buzón
$boxInfo:=$transporter.selectBox("Inbox")

//obtener el objeto Email con ID 1
$mail:=$transporter.getMail(1)

```

## .getMails()

► Histórico

`.getMails( ids : Collection { ; options : Object } ) : Object`  
`.getMails( startMsg : Integer ; endMsg : Integer { ; options : Object } ) : Object`

Parámetros	Tipo		Descripción
ids	Collection	->	Colección de identificadores de mensajes
startMsg	Integer	->	Número de secuencia del primer mensaje
endMsg	Integer	->	Número de secuencia del último mensaje
options	Objeto	->	Instrucciones sobre el manejo de mensajes
Resultado	Objeto	<-	Objeto que contiene: <ul style="list-style-type: none"> <li>una colección de <a href="#">objetos Email</a> y</li> <li>una colección de identificadores o números para los mensajes que faltan, si los hay</li> </ul>

### Descripción

La función `.getMails()` devuelve un objeto que contiene una colección de objetos [Email](#).

Primera sintaxis:

`.getMails( ids { ; options } ) -> result`

La primera sintaxis permite recuperar los mensajes en función de sus identificadores.

En el parámetro `ids`, pase una colección de IDs para los mensajes a devolver. Puede obtener los IDs con [.getMail\(\)](#).

El parámetro opcional `options` permite definir las partes de los mensajes a devolver. Consulte la tabla [Opciones](#) a continuación para obtener una descripción de las propiedades disponibles.

Segunda sintaxis:

`.getMails( startMsg ; endMsg { ; options } ) -> result`

La segunda sintaxis permite recuperar los mensajes en función de un rango secuencial. Los valores pasados representan la posición de los mensajes en el buzón.

En el parámetro `startMsg`, pase un valor [entero](#) correspondiente al número del primer mensaje en un rango secuencial. Si se pasa un número negativo (`startMsg <= 0`), se utilizará el primer mensaje del buzón como inicio de la secuencia.

En el parámetro `endMsg`, pase un valor *entero* correspondiente al número del último mensaje a incluir en un rango secuencial. Si se pasa un número negativo (`endMsg <= 0`), se utilizará el último mensaje del buzón como fin de secuencia.

El parámetro opcional `options` permite definir las partes de los mensajes a devolver.

#### Opciones

Propiedad	Tipo	Descripción
<code>updateSeen</code>	Booleano	Si True, los mensajes especificados se marcan como "vistos" en el buzón. Si False, los mensajes no se marcan como "vistos". Valor por defecto: True
<code>withBody</code>	Booleano	Pase True para devolver el cuerpo de los mensajes específicos. Si False, sólo se devuelve los encabezados de los mensajes. Valor por defecto: True

- Si no se selecciona ningún buzón con el comando `.selectBox()`, se genera un error.
- Si no hay ninguna conexión abierta, `.getMails()` abrirá una conexión con el último buzón especificado por `.selectBox()`.

#### Resultado

`.getMails()` devuelve un objeto que contiene las siguientes colecciones:

Propiedad	Tipo	Descripción
<code>lista</code>	Collection	Colección de objetos <code>Email</code> . Si no se encuentran objetos Email, se devuelve una colección vacía.
<code>notFound</code>	Collection	Colección de: <ul style="list-style-type: none"><li>• primera sintaxis - IDs de mensajes pasados previamente que no existen</li><li>• segunda sintaxis - números de secuencia de los mensajes entre <code>startMsg</code> y <code>endMsg</code> que no existen</li></ul> Se devuelve una colección vacía si se encuentran todos los mensajes.

#### Ejemplo

Quiere recuperar los 20 correos electrónicos más recientes sin cambiar el estado "visto":

```

var $server,$boxInfo,$result : Object
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com" //Obligatorio
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

//crear transportador
$transporter:=IMAP New transporter($server)

//seleccionar buzón
$boxInfo:=$transporter.selectBox("INBOX")

If($boxInfo.mailCount>0)
    // recuperar los encabezados de los últimos 20 mensajes sin marcarlos como leídos
    $result:=$transporter.getMails($boxInfo.mailCount-20;$boxInfo.mailCount;\ 
        New object("withBody";False;"updateSeen";False))
    For each($mail;$result.list)
        // ...
    End for each
End if

```

## .getMIMEAsBlob()

► Histórico

.getMIMEAsBlob( *msgNumber* : Integer { ; *updateSeen* : Boolean } ) : Blob

.getMIMEAsBlob( *msgID* : Text { ; *updateSeen* : Boolean } ) : Blob

Parámetros	Tipo		Descripción
<i>msgNumber</i>	Integer	->	Número de secuencia del mensaje
<i>msgID</i>	Texto	->	ID único del mensaje
<i>updateSeen</i>	Booleano	->	Si True, el mensaje se marca como "visto" en el buzón. Si False, el mensaje se deja igual.
Resultado	BLOB	<-	Blob de la cadena MIME devuelta por el servidor de correo

### Descripción

La función `.getMIMEAsBlob()` devuelve un BLOB con el contenido MIME del mensaje correspondiente al *msgNumber* o *msgID* en el buzón designado por el `IMAP_transporter`.

En el primer parámetro, puede pasar:

- *msgNumber*, un valor *integer* indicando el número de secuencia del mensaje a recuperar o
- *msgID*, un valor *text* indicando el ID único del mensaje a recuperar.

El parámetro opcional *updateSeen* permite indicar si el mensaje está marcado como "visto" en el buzón. Puede pasar:

- True - para marcar el mensaje como "visto" (indicando que el mensaje ha sido leído)
- False - para dejar intacto el estado "visto" del mensaje > \* La función devuelve un BLOB vacío si *msgNumber* o *msgID\** designa un mensaje inexistente, > \* Si no se selecciona ningún buzón con el comando `.selectBox()`, se genera un error, > \* Si no hay ninguna conexión abierta, `.getMIMEAsBlob()` abrirá una conexión con el último buzón especificado por `.selectBox()` .

- La función devuelve un BLOB vacío si *msgNumber* o *msgID\** designa un mensaje inexistente,
- Si no se selecciona ningún buzón con el comando `.selectBox()` , se genera un error,
- Si no hay ninguna conexión abierta, `.getMIMEAsBlob()` abrirá una conexión con el último buzón

especificado por `.selectBox()`.

## Resultado

`.getMIMEAsBlob()` devuelve un `BLOB` que puede almacenarse en una base de datos o convertirse en un objeto [Email](#) con el comando `MAIL Convert from MIME`.

## Ejemplo

```
var $server : Object
var $boxInfo : Variant
var $blob : Blob
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com"
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXXX"

//crear transportador
$transporter:=IMAP New transporter($server)

//seleccionar buzón
$boxInfo:=$transporter.selectBox("Inbox")

//obtener BLOB
$blob:=$transporter.getMIMEAsBlob(1)
```

## .host

► Histórico

.host : Text

### Descripción

La propiedad `.host` contiene el nombre o la dirección IP del servidor local. Se utiliza para las transacciones de correo (SMTP, POP3, IMAP).

## .logFile

► Histórico

.logFile : Text

### Descripción

La propiedad `.logFile` contiene la ruta del archivo de registro extendido definido (si existe) para la conexión de correo. Puede ser relativo (a la carpeta actual Logs) o absoluto.

A diferencia de los archivos de registro clásicos (habilitados mediante el comando `SET DATABASE PARAMETER`), los archivos de registro extendidos almacenan el contenido MIME de todos los correos enviados y no tienen ningún límite de tamaño. Para más información sobre los archivos de registro extendidos, consulte:

- Conexiones SMTP - [4DSMTPLLog.txt](#)
- \*\*Conexiones POP3 \*\* - [4DSMTPLLog.txt](#)
- Conexiones IMAP - [4DIMAPLog.txt](#)

## .move()

► Histórico

.move( *msgsIDs* : Collection ; *destinationBox* : Text ) : Object

.move( *allMsgs* : Integer ; *destinationBox* : Text ) : Object

Parámetros	Tipo		Descripción
<i>msgsIDs</i>	Collection	->	Colección de identificadores únicos de mensajes (cadenas)
<i>allMsgs</i>	Integer	->	IMAP all : todos los mensajes del buzón seleccionado
<i>destinationBox</i>	Texto	->	Buzón para recibir los mensajes desplazados
Resultado	Objeto	<-	Estado de la operación de desplazamiento

## Descripción

La función `move()` mueve los mensajes definidos en *msgsIDs* o *allMsgs* a la *destinationBox* en el servidor IMAP.

Puede pasar:

- en el parámetro *msgsIDs*, una colección contiene los IDs únicos de los mensajes específicos a mover,
- en el parámetro *allMsgs*, la constante `IMAP all` (entero) para mover todos los mensajes del buzón seleccionado.

El parámetro *destinationBox* permite pasar un valor texto con el nombre del buzón donde los mensajes serán desplazados.

Esta función sólo la soportan los servidores IMAP que cumplen con el RFC [8474](#).

## Objeto devuelto

La función devuelve un objeto que describe el estado IMAP:

Propiedad		Tipo	Descripción
<i>success</i>		Booleano	True si la operación tiene éxito, False en caso contrario
<i>statusText</i>		Texto	Mensaje de estado devuelto por el servidor IMAP, o último error devuelto en la pila de errores de 4D
<i>errors</i>		Collection	Pila de errores 4D (no se devuelve si se recibe una respuesta del servidor IMAP)
	[].errcode	Número	Código de error 4D
	[].message	Texto	Descripción del error 4D
	[].componentSignature	Texto	Firma del componente interno que ha devuelto el error

## Ejemplo 1

Para mover una selección de mensajes:

```

var $server;$boxInfo;$status : Object
var $mailIds : Collection
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com" //Obligatorio
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

$transporter:=IMAP New transporter($server)

//seleccionar buzón
$boxInfo:=$transporter.selectBox("inbox")

//obtener la colección de IDs únicos de los mensajes
$mailIds:=$transporter.searchMails("subject \\"4D new feature:\\"")

// Mover los mensajes encontrados del buzón actual al buzón "documents"
$status:=$transporter.move($mailIds;"documents")

```

## Ejemplo 2

Para mover todos los mensajes del buzón actual:

```

var $server;$boxInfo;$status : Object
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com" //Obligatorio
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

$transporter:=IMAP New transporter($server)

//seleccionar buzón
$boxInfo:=$transporter.selectBox("inbox")

// mover todos los mensajes del buzón actual al buzón "documents"
$status:=$transporter.move(IMAP all;"documents")

```

## .numToID()

► Histórico

.numToID( *startMsg* : Integer ; *endMsg* : Integer ) : Collection

Parámetros	Tipo		Descripción
<i>startMsg</i>	Integer	->	Número de secuencia del primer mensaje
<i>endMsg</i>	Integer	->	Número de secuencia del último mensaje
Resultado	Collection	<-	Colección de identificadores de mensajes únicos

### Descripción

La función `.numToID()` convierte los números de secuencia en IDs únicos IMAP para los mensajes en el rango secuencial designado por *startMsg* y *endMsg* en el buzón actualmente seleccionado.

En el parámetro *startMsg*, pase un valor entero correspondiente al número del primer mensaje en un rango secuencial.

Si se pasa un número negativo (`startMsg <= 0`), se utilizará el primer mensaje del buzón como inicio de la secuencia.

En el parámetro `endMsg`, pase un valor entero correspondiente al número del último mensaje a incluir en un rango secuencial. Si se pasa un número negativo (`endMsg <= 0`), se utilizará el último mensaje del buzón como fin de secuencia.

## Resultado

La función devuelve una colección de cadenas (IDs únicos).

## Ejemplo

```
var $transporter : 4D.IMAPTransporter
var $server;$boxInfo;$status : Object
var $mailIds : Collection

$server:=New object
$server.host:="imap.gmail.com" //Obligatorio
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXXX"

$transporter:=IMAP New transporter($server)

//seleccionar buzón
$boxInfo:=$transporter.selectBox("inbox")

//obtener los ID de los 5 últimos mensajes recibidos
$mailIds:=$transporter.numToID(($boxInfo.mailCount-5);$boxInfo.mailCount)

//eliminar los mensajes del buzón actual
$status:=$transporter.delete($mailIds)
```

## .removeFlags()

► Histórico

`.removeFlags( msgIDs : Collection ; keywords : Object ) : Object`

`.removeFlags( msgIDs : Text ; keywords : Object ) : Object`

`.removeFlags( msgIDs : Longint ; keywords : Object ) : Object`

Parámetros	Tipo		Descripción
msgIDs	Collection	->	Colección de cadenas: identificadores únicos de mensajes (texto) Texto: ID único de un mensaje Longint (IMAP all): todos los mensajes del buzón seleccionado
keywords	Objeto	->	Banderas de palabras claves a eliminar
Resultado	Objeto	<-	Estado de la operación removeFlags

## Descripción

La función `.removeFlags()` elimina las banderas de los `msgIDs` para las `keywords` definidas.

En el parámetro `msgIDs`, puede pasar:

- una *collection* contiene los identificadores únicos de mensajes específicos, o
- el ID único (*texte*) de un solo mensaje o
- la siguiente constante (*longint*) para todos los mensajes del buzón seleccionado:

Constante	Valor	Comentario
IMAP all	1	Seleccionar todos los mensajes del buzón seleccionado

El parámetro `keywords` le permite pasar un objeto con valores de palabras clave para las banderas específicas a eliminar de los `msgIDs`. Puede pasar cualquiera de las siguientes palabras claves:

Parámetros	Tipo	Descripción
\$draft	Booleano	True para eliminar el marcador "draft" del mensaje
\$seen	Booleano	True para eliminar el marcador "seen" del mensaje
\$flagged	Booleano	True para eliminar el marcador "flagged" del mensaje
\$answered	Booleano	True para eliminar el marcador "answered" del mensaje
\$deleted	Booleano	True para eliminar el marcador "deleted" del mensaje

Note que los valores falsos se ignoran.

Objeto devuelto

La función devuelve un objeto que describe el estado IMAP:

Propiedad		Tipo	Descripción
success		Booleano	True si la operación tiene éxito, False en caso contrario
statusText		Texto	Mensaje de estado devuelto por el servidor IMAP, o último error devuelto en la pila de errores de 4D
errors		Collection	Pila de errores 4D (no se devuelve si se recibe una respuesta del servidor IMAP)
	[].errcode	Número	Código de error 4D
	[].message	Texto	Descripción del error 4D
	[].componentSignature	Texto	Firma del componente interno que ha devuelto el error

Ejemplo

```
var $options;$transporter;$boxInfo;$status : Object

$options:=New object
$options.host:="imap.gmail.com"
$options.port:=993
$options.user:="4d@gmail.com"
$options.password:="xxxxx"

// Crear transportador
$transporter:=IMAP New transporter($options)

// Seleccionar buzón
$boxInfo:=$transporter.selectBox("INBOX")

// Marcar todos los mensajes de INBOX como no vistos
$flags:=New object
$flags["$seen"]:=True
$status:=$transporter.removeFlags(IMAP all;$flags)
```

## .renameBox()

► Histórico

.renameBox( *currentName* : Text ; *newName* : Text ) : Object

Parámetros	Tipo		Descripción
<i>currentName</i>	Texto	->	Nombre del nuevo actual
<i>newName</i>	Texto	->	Nuevo nombre del buzón
Resultado	Objeto	<-	Estado de la operación renombrar

## Descripción

La función `.renameBox()` cambia el nombre de un buzón en el servidor IMAP. Si se intenta renombrar un buzón desde un nombre de buzón que no existe o a un nombre de buzón que ya existe, se generará un error.

En el parámetro `currentName`, pase el nombre del buzón a renombrar.

Pase el nuevo nombre del buzón en el parámetro `newName`.

## Objeto devuelto

La función devuelve un objeto que describe el estado IMAP:

Propiedad		Tipo	Descripción
<code>success</code>		Booleano	True si la operación tiene éxito, False en caso contrario
<code>statusText</code>		Texto	Mensaje de estado devuelto por el servidor IMAP, o último error devuelto en la pila de errores de 4D
<code>errors</code>		Collection	Pila de errores 4D (no se devuelve si se recibe una respuesta del servidor IMAP)
	<code>[] .errcode</code>	Número	Código de error 4D
	<code>[] .message</code>	Texto	Descripción del error 4D
	<code>[] .componentSignature</code>	Texto	Firma del componente interno que ha devuelto el error

## Ejemplo

Para cambiar el nombre de su buzón de "Invoices" a "Bills":

```
var $pw : text
var $options; $transporter; $status : object

$options:=New object

$pw:=Request("Please enter your password:")

If(OK=1) $options.host:="imap.gmail.com"
$options.user:="test@gmail.com"
$options.password:=$pw

$transporter:=IMAP New transporter($options)

// renombrar buzón
$status:=$transporter.renameBox("Invoices"; "Bills")

If ($status.success)
  ALERT("Mailbox renaming successful!")
Else
  ALERT("Error: "+$status.statusText)
End if
End if
```

## .port

► Histórico

.port : Integer

### Descripción

La propiedad `.port` contiene el número de puerto utilizado para las transacciones de correo. Por defecto, si la propiedad `port` no se ha definido en el objeto `server` (utilizado para crear el objeto transportador con `SMTP New transporter`, `POP3 New transporter`, `IMAP New transporter`), el puerto utilizado es:

- SMTP - 587
- POP3 - 995
- IMAP - 993

## .searchMails()

► Histórico

.searchMails( *searchCriteria* : Text ) : Collection

Parámetros	Tipo		Descripción
searchCriteria	Texto	->	Criterio de búsqueda
Resultado	Collection	<-	Colección de números de mensajes

### Descripción

Esta función se basa en la especificación del [protocolo IMAP](#).

La función `.searchMails()` busca los mensajes que coincidan con el `searchCriteria` en el buzón actual. El parámetro `searchCriteria` contiene una o varias palabras clave de búsqueda.

`searchCriteria` es un parámetro texto que enumera una o varias llaves de búsqueda (ver [llaves-de-búsqueda-autorizadas](#) más abajo) asociadas o no a valores a buscar. Una llave de búsqueda puede ser uno o varios elementos. Por ejemplo:

```
SearchKey1 = FLAGGED  
SearchKey2 = NOT FLAGGED  
SearchKey3 = FLAGGED DRAFT
```

Las coincidencias no suelen diferenciar entre mayúsculas y minúsculas

- Si el `searchCriteria` es una cadena null, la búsqueda será equivalente a un "seleccionar todo".
- Si `searchCriteria` incluye varias llaves de búsqueda, el resultado es la intersección (función AND) de todos los mensajes que coinciden con esas llaves.

```
searchCriteria = FLAGGED FROM "SMITH"
```

... devuelve todos los mensajes con el marcador \$Flagged AND activado y enviados por Smith.

- Puede utilizar los operadores OR o NOT de la siguiente manera:

```
searchCriteria = OR SEEN FLAGGED
```

... devuelve todos los mensajes con el marcador \$Seen O \$Flagged

```
searchCriteria = NOT SEEN
```

... devuelve todos los mensajes con el marcador \$Seen.

```
searchCriteria = HEADER CONTENT-TYPE "MIXED" NOT HEADER CONTENT-TYPE "TEXT" ...
```

... devuelve los mensajes cuyo encabezado content-type contiene "Mixed" y no contiene "Text".

```
searchCriteria = HEADER CONTENT-TYPE "E" NOT SUBJECT "o" NOT HEADER CONTENT-TYPE "MIXED"
```

... devuelve los mensajes cuyo encabezado content-type contiene " e " y cuyo encabezado Subject no contiene " o " y cuyo encabezado content-type no es " Mixed ".

En cuanto a los dos últimos ejemplos, observe que el resultado de la búsqueda es diferente cuando se eliminan los paréntesis de la primera lista de llaves de búsqueda.

- El parámetro `searchCriteria` puede incluir opcionalmente la instrucción [CHARSET]. Esta instrucción consiste en la palabra "CHARSET" seguida de un conjunto de caracteres definido [CHARSET] (US ASCII, ISO-8859). Indica el conjunto de caracteres de la cadena `searchCriteria`. Por lo tanto, debe convertir la cadena `searchCriteria` en el conjunto de caracteres especificado si utiliza la instrucción [CHARSET] (ver los comandos `CONVERT FROM TEXT` o `Convert to text`). Por defecto, 4D codifica la cadena de criterios `searchCriteria` en Quotable Printable si contiene los caracteres extendidos.

```
searchCriteria = CHARSET "ISO-8859" BODY "Help"
```

... significa que los criterios de búsqueda utilizan el conjunto de caracteres iso-8859 y el servidor tendrá que convertir los criterios de búsqueda antes de buscar, si es necesario.

## Tipos de valores de búsqueda

Las claves de búsqueda pueden solicitar el valor a buscar:

- Valores de tipo fecha: los valores de tipo fecha se colocan en cadenas con el siguiente formato: `date-day+-"+date-month+-"+date-year` donde date-day indica la fecha del día del mes (2 caracteres como máximo), date-month indica el mes (Jan/Feb/Mar/Apr/May/Jun/Jul/Aug/Sep/Oct/Dec) y date-year indica el año (4 dígitos).  
Ejemplo: `searchCriteria = SENTBEFORE 1-Feb-2020` (una fecha no suele necesitar comillas, ya que no contiene caracteres especiales)
- Valores de tipo cadena: la cadena puede contener cualquier carácter y debe ir entre comillas. Si la cadena no contiene ningún carácter especial, como el espacio, por ejemplo, no es necesario colocarla entre comillas. Al colocar entre comillas estas cadenas se garantiza que su valor se interpretará correctamente. Ejemplo: `criterios de búsqueda = FROM "SMITH"` Para todas las llaves de búsqueda que utilizan cadenas, un mensaje coincide con la llave si la cadena es una subcadena del campo. Las coincidencias no diferencian entre mayúsculas y minúsculas.
- Search-keys with a flag value: the flag may accept one or several keywords (including standard flags), separated by spaces. Example: `searchCriteria = HEADER CONTENT-TYPE "MIXED"`
- Marcadores: los valores de tipo marcador (flags) aceptan una o varias palabras claves (incluyendo marcadores estándar) separados por espacios. Ejemplo: `searchCriteria = KEYWORD \Flagged \Draft`
- Conjunto de mensajes: identifica un conjunto de mensajes. En el caso de los números de secuencia de los mensajes, se trata de números consecutivos desde el 1 hasta el número total de mensajes en el buzón. Los números son separados por coma; un dos puntos (:) delimita entre dos números inclusive. Ejemplos:  
`2,4:7,9,12:*` es `2,4,5,6,7,9,12,13,14,15` para un buzón con 15 mensajes. `searchCriteria = 1:5 ANSWERED` busca en la selección de mensajes 1 a 5, los mensajes que tienen el marcador \$Answered. `searchCriteria= 2,4 ANSWERED` busca en la selección de mensajes (números de mensaje 2 y 4) los mensajes que tienen el marcador \$Answered.

## Teclas de búsqueda disponibles

ALL: todos los mensajes del 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 que tienen el marcador \$Recent.  
OLD: mensajes que no tienen el marcador \$Recent.  
SEEN: mensajes que tienen el marcador \$Seen.  
UNSEEN: Mensajes que no tienen el marcador \$Seen.  
NEW: mensajes que tienen el marcador \$Recent pero no el marcador \$Seen. Esto es equivalente a "(RECENT UNSEEN)".  
\*\*\*KEYWORD \*\*\*flag\*\*\*\*\*: mensajes con el conjunto de palabras clave especificado.  
\*\*\*UNKEYWORD \*\*\*flag\*\*\*\*\*: mensajes que no tienen la palabra clave especificada.  
\*\*\*BEFORE \*\*\*date\*\*\*\*\*: mensajes cuya fecha interna es anterior a la fecha especificada.  
\*\*\*ON \*\*\*date\*\*\*\*\*: mensajes cuya fecha interna está dentro de la fecha especificada.  
\*\*\*SINCE \*\*\*date\*\*\*\*\*: mensajes cuya fecha interna está dentro o es posterior a la fecha especificada.  
\*\*\*SENTBEFORE \*\*\*date\*\*\*\*\*: mensajes cuyo encabezado Date es anterior a la fecha especificada.  
\*\*\*SENTON \*\*\*date\*\*\*\*\*: mensajes cuyo encabezado Date está dentro de la fecha especificada.  
\*\*\*SENTSINCE \*\*\*date\*\*\*\*\*: mensajes cuyo encabezado Date está dentro o es posterior a la fecha especificada.  
\*\*\*TO \*\*\*string\*\*\*\*\*: mensajes que contienen la cadena especificada en el encabezado TO.  
\*\*\*FROM \*\*\*string\*\*\*\*\*: mensajes que contienen la cadena especificada en el encabezado FROM.  
\*\*\*CC \*\*\*string\*\*\*\*\*: mensajes que contienen la cadena especificada en el encabezado CC.  
\*\*\*BCC \*\*\*string\*\*\*\*\*: mensajes que contienen la cadena especificada en el encabezado BCC.  
\*\*\*SUBJECT \*\*\*string\*\*\*\*\*: mensajes que contienen la cadena especificada en el encabezado Asunto.  
\*\*\*BODY \*\*\*string\*\*\*\*\*: mensajes que contienen la cadena especificada en el cuerpo del mensaje.  
\*\*\*TEXT \*\*\*string\*\*\*\*\*: mensajes que contienen la cadena especificada en el encabezado o en el cuerpo del mensaje.  
\*\*\*HEADER *field-name* \*\*\*string\*\*\*\*\*: mensajes que tienen un encabezado con el field-name especificado y que contienen la cadena especificada en el field-body.  
\*\*\*UID \*\*\*message-UID\*\*\*\*\*: mensajes con identificadores únicos correspondientes al conjunto de identificadores únicos especificado.  
\*\*\*LARGER \*\*\*n\*\*\*\*\*: mensajes con un tamaño superior al número de bytes especificado.  
\*\*\*SMALLER \*\*\*n\*\*\*\*\*: mensajes con un tamaño inferior al número de bytes especificado.  
\*\*\*NOT \*\*\*search-key\*\*\*\*\*: mensajes que no coinciden con el criterio de búsqueda especificado.  
\*\*\*OR *search-key1* \*\*\**search-key2*\*\*\*\*\*: mensajes que coinciden con cualquiera de las dos criterios de búsqueda.

## .selectBox()

► Histórico

.selectBox( *name* : Text { ; *state* : Integer } ) : Object

Parámetros	Tipo		Descripción
<i>name</i>	Texto	->	Nombre del buzón
<i>state</i>	Integer	->	Estado de acceso al buzón
Resultado	Objeto	<-	objeto boxInfo

### Descripción

La función `.selectBox()` selecciona el buzón *name* como el buzón actual. Esta función permite recuperar la información sobre el buzón.

Para obtener la información de un buzón sin cambiar el buzón actual, utilice `.getBoxInfo()`.

En el parámetro `name`, pase el nombre del buzón a acceder. El nombre representa una jerarquía inequívoca de izquierda a derecha, con niveles separados por un carácter delimitador específico. El delimitador se puede recuperar con la función `.getDelimiter()`.

El parámetro opcional `state` define el tipo de acceso al buzón. Los valores posibles son:

Constante	Valor	Comentario
IMAP read only state	1	Se accede al buzón seleccionado con privilegios de sólo lectura. Los mensajes con la bandera "reciente" (que indica que son nuevos) no se modifican.
IMAP read write state	0	Se accede al buzón seleccionado con privilegios de lectura y escritura. Los mensajes se consideran "vistos" y pierden la bandera "reciente" (que indica que son mensajes nuevos). (Valor por defecto)

- La función genera un error y devuelve Null si `name` designa un buzón inexistente.
- Si no hay ninguna conexión abierta, `.selectBox()` abrirá una conexión.
- Si la conexión no se ha utilizado desde el retraso de conexión designado (ver `IMAP New transporter`), se llama automáticamente a la función `.checkConnection()`.

#### Objeto devuelto

El objeto `boxInfo` devuelto contiene las siguientes propiedades:

Propiedad	Tipo	Descripción
<code>name</code>	Texto	Nombre del buzón
<code>mailCount</code>	number	Número de mensajes en el buzón
<code>mailRecent</code>	number	Número de mensajes con la bandera "recent"

#### Ejemplo

```
var $server; $boxinfo : Object
$server:=New object
$server.host:="imap.gmail.com" //Mandatory
$server.user:="4d@gmail.com"
$server.password:="XXXXXXXX"

var $transporter : 4D.IMAPTransporter
$transporter:=IMAP New transporter($server)
$boxInfo:=$transporter.selectBox("INBOX")
```

## .subscribe()

► Histórico

`.subscribe( name : Text ) : Object`

Parámetros	Tipo		Descripción
<code>name</code>	Texto	->	Nombre del buzón
Resultado	Objeto	<-	Estado de la operación subscribe

#### Descripción

La función `.subscribe()` permite añadir o eliminar el buzón especificado del conjunto de buzones "suscritos" en el servidor IMAP. De este modo, puede optar por acotar una gran lista de buzones disponibles suscribiéndose a los que habitualmente consulta.

En el parámetro `name`, pase el nombre del buzón a añadir (suscribir) a sus buzones "suscritos".

#### Objeto devuelto

La función devuelve un objeto que describe el estado IMAP:

Propiedad		Tipo	Descripción
success		Booleano	True si la operación tiene éxito, False en caso contrario
statusText		Texto	Mensaje de estado devuelto por el servidor IMAP, o último error devuelto en la pila de errores de 4D
errors		Collection	Pila de errores 4D (no se devuelve si se recibe una respuesta del servidor IMAP)
	[].errcode	Número	Código de error 4D
	[].message	Texto	Descripción del error 4D
	[].componentSignature	Texto	Firma del componente interno que ha devuelto el error

#### Ejemplo

Para suscribirse al buzón "Atlas Corp" en la jerarquía "Bills":

```
var $pw; $name : text
var $options; $transporter; $status : object

$options:=New object

$pw:=Request("Please enter your password:")

If(OK=1) $options.host:="imap.gmail.com"
$options.user:="test@gmail.com"
$options.password:=$pw

$transporter:=IMAP New transporter($options)

$name:="Bills"+$transporter.getDelimiter()+"Atlas Corp"
$status:=$transporter.subscribe($name)

If ($status.success)
  ALERT("Mailbox subscription successful!")
Else
  ALERT("Error: "+$status.statusText)
End if
End if
```

## .unsubscribe()

► Histórico

.unsubscribe( *name* : Text ) : Object

Parámetros	Tipo		Descripción
name	Texto	->	Nombre del buzón
Resultado	Objeto	<-	Estado de la operación unsubscribe

#### Descripción

La función `.unsubscribe()` elimina un buzón de un conjunto de buzones suscritos. Esto le permite reducir el número de buzones que suele ver.

En el parámetro `name`, pase el nombre del buzón a eliminar (darse de baja) de sus buzones activos.

#### Objeto devuelto

La función devuelve un objeto que describe el estado IMAP:

Propiedad		Tipo	Descripción
success		Booleano	True si la operación tiene éxito, False en caso contrario
statusText		Texto	Mensaje de estado devuelto por el servidor IMAP, o último error devuelto en la pila de errores de 4D
errors		Collection	Pila de errores 4D (no se devuelve si se recibe una respuesta del servidor IMAP)
	[].errcode	Número	Código de error 4D
	[].message	Texto	Descripción del error 4D
	[].componentSignature	Texto	Firma del componente interno que ha devuelto el error

#### Ejemplo

Para desuscribirse del buzón "Atlas Corp" en la jerarquía "Bills":

```
var $pw; $name : text
var $options; $transporter; $status : object

$options:=New object

$pw:=Request("Please enter your password:")

If(OK=1) $options.host:="imap.gmail.com"
$options.user:="test@gmail.com"
$options.password:=$pw

$transporter:=IMAP New transporter($options)

$name:="Bills"+$transporter.getDelimiter()+"Atlas Corp"
$status:=$transporter.unsubscribe($name)

If ($status.success)
  ALERT("Mailbox unsubscribe successful!")
Else
  ALERT("Error: "+$status.statusText)
End if
End if
```

#### .user

► Histórico

.user : Text

#### Descripción

La propiedad `.user` contiene el nombre de usuario utilizado para la autenticación en el servidor de correo.

# MailAttachment

Los objetos adjuntos permiten referenciar archivos en un objeto `Email`. Los objetos Attachment (adjuntos) son creados utilizando el comando `MAIL New attachment`.

## Objetos adjuntos

Los objetos Attachment ofrecen las siguientes propiedades y funciones de sólo lectura:

<code>.cid : Text</code>	el ID del adjunto
<code>.disposition : Text</code>	el valor del encabezado <code>Content-Disposition</code>
<code>.getContent() : 4D.Blob</code>	devuelve el contenido del objeto adjunto en un objeto <code>4D.Blob</code>
<code>.name : Text</code>	el nombre y la extensión del adjunto
<code>.path : Text</code>	la ruta POSIX del archivo adjunto, si existe
<code>.platformPath : Text</code>	la ruta del archivo adjunto expresada con la sintaxis de la plataforma actual
<code>.type : Text</code>	el <code>content-type</code> del archivo adjunto

## MAIL New attachment

► Histórico

```
MAIL New attachment( file : 4D.File { ; name : Text {; cid : Text{ ; type : Text { ; disposition :Text } } } } ) :  
4D.MailAttachment  
MAIL New attachment( zipFile : 4D.ZipFile { ; name : Text {; cid : Text{ ; type : Text { ; disposition :Text } } } } ) :  
4D.MailAttachment  
MAIL New attachment( blob : 4D.Blob { ; name : Text {; cid : Text{ ; type : Text { ; disposition :Text } } } } ) :  
4D.MailAttachment  
MAIL New attachment( path : Text { ; name : Text {; cid : Text{ ; type : Text { ; disposition :Text } } } } ) :  
4D.MailAttachment
```

Parámetros	Tipo		Descripción
file	4D.File	->	Archivo adjunto
zIPFile	4D.ZipFile	->	Archivo zip adjunto
blob	4D.Blob	->	BLOB que contiene el archivo adjunto
path	Texto	->	Ruta del archivo adjunto
name	Texto	->	Nombre + extensión utilizados por el cliente de correo para designar el archivo adjunto
cid	Texto	->	ID del archivo adjunto (sólo en mensajes HTML), o " " si no se requiere cid
type	Texto	->	Valor del encabezado content-type
disposition	Texto	->	Valor del encabezado content-disposition: "inline" o "attachment".
Resultado	4D.MailAttachment	<-	Objeto adjunto

## Descripción

El comando `MAIL New attachment` le permite crear un objeto adjunto que puede asociar a un objeto [Email](#).

Para definir el adjunto, puede utilizar:

- un `file`, pase un objeto `4D.File` que contenga el propio archivo adjunto.
- un `zipfile`, pase un objeto `4D.ZipFile` que contenga el propio archivo adjunto.
- un `blob`, pase un objeto `4D.Blob` que contenga el propio archivo adjunto.
- un `path`, pase un valor texto que contenga la ruta del archivo adjunto, expresada con la sintaxis del sistema. Puede pasar un nombre de ruta completo o un simple nombre de archivo (en cuyo caso 4D buscará el archivo en el mismo directorio que el archivo del proyecto).

El parámetro opcional `name` permite pasar el nombre y la extensión que utilizará el cliente de correo para designar el archivo adjunto. Si se omite `name` y:

- pasó una ruta de archivo, se utiliza el nombre y la extensión del archivo,
- pasó un BLOB, se genera automáticamente un nombre aleatorio sin extensión.

El parámetro opcional `cid` permite pasar un ID interno para el archivo adjunto. Este ID es el valor del encabezado `Content-Id`, se utilizará sólo en mensajes HTML. El `cid` asocia el archivo adjunto con una referencia definida en el cuerpo del mensaje mediante una etiqueta HTML como `\`. Esto significa que el contenido del archivo adjunto (por ejemplo, una imagen) debe mostrarse dentro del mensaje en el cliente de correo. El resultado final puede variar en función del cliente de correo. Puede pasar una cadena vacía en `cid` si no quiere utilizar este parámetro.

Puede utilizar el parámetro opcional `type` para definir explícitamente el `content-type` del archivo adjunto. Por ejemplo, puede pasar una cadena que defina un tipo MIME ("video/mpeg"). Este valor de `content-type` se definirá para el archivo adjunto, independientemente de su extensión. Para más información sobre los tipos MIME, consulte la página [Tipo MIME en Wikipedia](#).

Por defecto, si el parámetro `type` se omite o contiene una cadena vacía, el `content-type` del archivo adjunto se basa en su extensión. Se aplican las siguientes reglas para los principales tipos MIME:

Extensión	Content-Type
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
otro	application/octet-stream

El parámetro opcional *disposition* permite pasar el encabezado `content-disposition` del adjunto. Puede pasar una de las siguientes constantes del tema constante "Mail":

Constante	Valor	Comentario
mail disposition attachment	"attachment"	Define el valor del encabezado Content-disposition como "attachment", lo que significa que el archivo adjunto debe proporcionarse como un enlace en el mensaje.
mail disposition inline	"inline"	Define el valor del encabezado Content-disposition como "inline", lo que significa que el archivo adjunto debe aparecer dentro del contenido del mensaje, en la ubicación "cid". La renderización depende del cliente de correo.

Por defecto, si se omite el parámetro *disposition*:

- si el parámetro *cid* se utiliza, el encabezado `Content-disposition` es definido como "inline",
- si el parámetro *cid* no se pasa o está vacío, el encabezado `Content-disposition` es definido como "attachment".

### Ejemplo 1

Desea enviar un correo electrónico con un archivo seleccionado por el usuario como adjunto y una imagen integrada en el cuerpo HTML:

```

$doc:=Select document("");%;"Please select a file to attach";0)
If (0K=1) //Si se ha seleccionado un documento

C_OBJECT($email;$server;$transporter)

$server:=New object
$server.host:="smtp.mail.com"
$server.user:="test_user@mail.com"
$server.password:="p@ssw@rd"
$transporter:=SMTP New transporter($server)

$email:=New object
$email.from:="test_user@mail.com"
$email.to:="test_user@mail.com"
$email.subject:="This is a test message with attachments"

//añadir un enlace para descargar el archivo
$email.attachments:=New collection(MAIL New attachment(Document))
//insertar una imagen en línea (utilice un cid)
$email.attachments[1]:=MAIL New attachment("c:\\\\Pictures\\\\4D.jpg";"";"4D")

$email.htmlBody:="<html>" + \
"<body>Hello World!" + \
"<img src='cid:4D' >" + \
"</body>" + \
"</head>" + \
"</html>"

$transporter.send($email) //enviar mail

```

**End if**

## Ejemplo 2

Desea enviar un correo electrónico con un área 4D Write Pro como archivo adjunto:

```

C_BLOB($blob)
WP EXPORT VARIABLE(WPArea;$blob;wk docx)

C_OBJECT($email;$server;$transporter)

$server:=New object
$server.host:="smtp.mail.com"
$server.user:="user@mail.com"
$server.password:="p@ssw@rd"
$transporter:=SMTP New transporter($server)

$email:=New object
$email.from:="user@mail.com"
$email.to:="customer@mail.com"
$email.subject:="New annual report"
$email.textBody:="Please find enclosed our latest annual report."
$email.attachments:=New collection(MAIL New attachment($blob;"Annual report.docx"))

$transporter.send($email)

```

## 4D.MailAttachment.new()

► Histórico

4D.MailAttachment.new( *file* : 4D.File { ; *name* : Text { ; *cid* : Text{ ; *type* : Text { ; *disposition* :Text } } } } ) :

```

4D.MailAttachment
4D.MailAttachment.new( zipFile : 4D.ZipFile { ; name : Text {; cid : Text{ ; type : Text { ; disposition :Text } } } } ) :
4D.MailAttachment
4D.MailAttachment.new( blob : 4D.Blob { ; name : Text {; cid : Text{ ; type : Text { ; disposition :Text } } } } ) :
4D.MailAttachment
4D.MailAttachment.new( path : Text { ; name : Text {; cid : Text{ ; type : Text { ; disposition :Text } } } } ) :
4D.MailAttachment

```

Parámetros	Tipo		Descripción
file	4D.File	->	Archivo adjunto
ZIPFile	4D.ZipFile	->	Archivo zip adjunto
blob	4D.Blob	->	BLOB que contiene el archivo adjunto
path	Texto	->	Ruta del archivo adjunto
name	Texto	->	Nombre + extensión utilizados por el cliente de correo para designar el archivo adjunto
cid	Texto	->	ID del archivo adjunto (sólo en mensajes HTML), o " " si no se requiere cid
type	Texto	->	Valor del encabezado content-type
disposition	Texto	->	Valor del encabezado content-disposition: "inline" o "attachment".
Resultado	4D.MailAttachment	<-	Objeto adjunto

## Descripción

La función `4D.MailAttachment.new()` crea y devuelve un nuevo objeto de tipo `4D.MailAttachment`. Es idéntico al comando `MAIL New attachment` (acceso directo).

### .cid

`.cid : Text`

## Descripción

La propiedad `.cid` contiene el ID del adjunto. Esta propiedad se utiliza sólo en los mensajes HTML. Si falta esta propiedad, el archivo se maneja como un simple adjunto (enlace).

### .disposition

`.disposition : Text`

## Descripción

La propiedad `.disposition` contiene el valor del encabezado `Content-Disposition`. Hay dos valores disponibles:

- "inline": el archivo adjunto se muestra dentro del contenido del mensaje, en la ubicación "cid". La renderización depende del cliente de correo.
- "attachment": el archivo adjunto se presenta como un enlace en el mensaje.

### .getContent()

`.getContent() : 4D.Blob`

Parámetros	Tipo		Descripción
Resultado	4D.Blob	<-	Contenido del anexo

## Descripción

La función `.getContent()` devuelve el contenido del objeto adjunto en un objeto `4D.Blob`. Puede utilizar esta función con los objetos adjuntos recibidos por el comando `MAIL Convert from MIME`.

## **.name**

`.name` : Text

## Descripción

La propiedad `.name` contiene el nombre y la extensión del adjunto. Por defecto, es el nombre del archivo, a menos que se haya indicado otro nombre en el comando `MAIL New attachment`.

## **.path**

`.path` : Text

## Descripción

La propiedad `.path` contiene la ruta POSIX del archivo adjunto, si existe.

## **.platformPath**

► Histórico

`.platformPath` : Text

## Descripción

La propiedad `.platformPath` contiene la ruta del archivo adjunto expresada con la sintaxis de la plataforma actual.

## **.type**

`.type` : Text

## Descripción

La propiedad `.type` contiene el `content-type` del archivo adjunto. Si este tipo no se pasa explícitamente al comando `MAIL New attachment`, el `content-type` se basa en su extensión de archivo.

# POP3Transporter

La clase `POP3Transporter` permite recuperar mensajes de un servidor de correo electrónico POP3.

## Objeto POP3 Transporter

Los objetos POP3 Transporter se instancian con el comando [POP3 New transporter](#). Ofrecen las siguientes propiedades y funciones:

<code>.acceptUnsecureConnection: Boolean</code>	True si se permite a 4D establecer una conexión no cifrada
<code>.authenticationMode: Text</code>	el modo de autenticación utilizado para abrir la sesión en el servidor de correo
<code>.checkConnection() : Object</code>	comprueba la conexión utilizando la información almacenada en el objeto transporter
<code>.connectionTimeOut : Integer</code>	el tiempo máximo de espera (en segundos) permitido para establecer una conexión con el servidor
<code>.delete( msgNumber : Integer )</code>	marca el email <i>msgNumber</i> a eliminar en el servidor POP3
<code>.getBoxInfo() : Object</code>	devuelve un objeto <code>boxInfo</code> correspondiente al buzón designado por el <code>Transporter</code>
<code>.getMail( msgNumber : Integer ) : Object</code>	devuelve el objeto <code>Email</code> correspondiente al <i>msgNumber</i> en el buzón designado por el <code>POP3 transporter</code>
<code>.getMailInfo( msgNumber : Integer ) : Object</code>	devuelve un objeto <code>mailInfo</code> correspondiente al <i>msgNumber</i> en el buzón designado por el <code>POP3 transporter</code>
<code>.getMailInfoList() : Collection</code>	devuelve una colección de objetos <code>mailInfo</code> que describen todos los mensajes del buzón designado por el <code>POP3 transporter</code>
<code>.getMIMEAsBlob( msgNumber : Integer ) : Blob</code>	devuelve un BLOB con el contenido MIME del mensaje correspondiente al <i>msgNumber</i> en el buzón designado por el <code>POP3 transporter</code>
<code>.host : Text</code>	el nombre o la dirección IP del servidor local
<code>.logFile : Text</code>	la ruta del archivo de registro extendido definido (si existe) para la conexión de correo
<code>.port : Integer</code>	el número de puerto utilizado para las transacciones de correo
<code>.undeleteAll()</code>	elimina todas las banderas de borrado definidas en los correos electrónicos en el <code>POP3_transporter</code>
<code>.user : Text</code>	el nombre de usuario utilizado para la autenticación en el servidor de correo

## POP3 New transporter

► Histórico

POP3 New transporter(*server* : Object) : 4D.POP3Transporter

Parámetros	Tipo		Descripción
server	objeto	->	Información del servidor de correo
Resultado	4D.POP3Transporter	<-	Objeto POP3 transporter

## Descripción

La función `.getMail()` devuelve el objeto `Email` correspondiente al `msgNumber` en el buzón designado por el [POP3 transporter](#). El objeto transportador devuelto se utilizará normalmente para recibir correos electrónicos.

En el parámetro `server`, pase un objeto que contenga las siguientes propiedades:

<code>server</code>	Valor por defecto (si se omite)
<code>.acceptUnsecureConnection: Boolean</code> True si se permite a 4D establecer una conexión no cifrada	False
<code>.accessTokenOAuth2: Text</code> <code>.accessTokenOAuth2: Object</code> Cadena de texto u objeto token que representan las credenciales de autorización OAuth 2. Sólo se utiliza con <code>OAUTH2 authenticationMode</code> . Si se utiliza <code>accessTokenOAuth2</code> pero se omite <code>authenticationMode</code> , se utiliza el protocolo OAuth 2 (si el servidor lo permite). No devuelto en el objeto <a href="#">SMTP transporter</a> .	ninguno
<code>.authenticationMode: Text</code> el modo de autenticación utilizado para abrir la sesión en el servidor de correo	se utiliza el modo de autenticación más seguro soportado por el servidor
<code>.connectionTimeOut : Integer</code> el tiempo máximo de espera (en segundos) permitido para establecer una conexión con el servidor	30
<code>.host : Text</code> el nombre o la dirección IP del servidor local	<i>obligatorio</i>
<code>.logFile : Text</code> la ruta del archivo de registro extendido definido (si existe) para la conexión de correo	ninguno
<code>.password : Text</code> Contraseña del usuario para la autenticación en el servidor. No devuelto en el objeto <a href="#">SMTP transporter</a> .	ninguno
<code>.port : Integer</code> el número de puerto utilizado para las transacciones de correo	995
<code>.user : Text</code> el nombre de usuario utilizado para la autenticación en el servidor de correo	ninguno

## Resultado

La función devuelve un [objeto POP3 transporter](#). Todas las propiedades devueltas son de sólo lectura.

La conexión POP3 se cierra automáticamente cuando se destruye el objeto transportador.

## Ejemplo

```

var $server : Object
$server:=New object
$server.host:="pop.gmail.com" //Obligatorio
$server.port:=995
$server.user:="4d@gmail.com"
$server.password:="XXXXXXXX"
$server.logFile:="LogTest.txt" //log a registrar en la carpeta Logs

var $transporter : 4D.POP3Transporter
$transporter:=POP3 New transporter($server)

$status:=$transporter.checkConnection()
If(Not($status.success))
    ALERT("An error occurred receiving the mail: "+$status.statusText)
End if

```

## 4D.POP3Transporter.new()

4D.POP3Transporter.new( *server* : Object ) : 4D.POP3Transporter

Parámetros	Tipo		Descripción
<i>server</i>	Objeto	->	Información del servidor de correo
Resultado	4D.POP3Transporter	<-	<a href="#">Objeto POP3 transporter</a>

### Descripción

La función `4D.POP3Transporter.new()` crea y devuelve un nuevo objeto del tipo `4D.POP3Transporter`. Es idéntico al comando `POP3 New transporter` (acceso directo).

## .acceptUnsecureConnection

► Histórico

`.acceptUnsecureConnection`: Boolean

### Descripción

La propiedad `.acceptUnsecureConnection` contiene True si se permite a 4D establecer una conexión no cifrada cuando la conexión cifrada no es posible.

Contiene False si no se permiten las conexiones no cifradas, en cuyo caso se devuelve un error cuando no es posible la conexión cifrada.

Los puertos seguros disponibles son:

- SMTP
  - 465: SMTPS
  - 587 o 25: SMTP con actualización STARTTLS si lo soporta el servidor.
- IMAP
  - 143: Puerto IMAP no encriptado
  - 993: IMAP con actualización STARTTLS si lo soporta el servidor
- POP3
  - 110: Puerto POP3 no encriptado
  - 995: POP3 con actualización STARTTLS si lo soporta el servidor.

## .authenticationMode

► Histórico

.authenticationMode: Text

### Descripción

La propiedad `.authenticationMode` contiene el modo de autenticación utilizado para abrir la sesión en el servidor de correo.

Por defecto, se utiliza el modo más seguro soportado por el servidor.

Los valores posibles son:

Valor	Constantes	Comentario
APOP	<code>POP3 authentication APOP</code>	Authentication using APOP protocol (POP3 only)
CRAM-MD5	<code>POP3 authentication CRAM-MD5</code>	Autenticación utilizando el protocolo CRAM-MD5
LOGIN	<code>POP3 authentication login</code>	Autenticación utilizando el protocolo LOGIN
OAuth2	<code>POP3 authentication OAuth2</code>	Autenticación utilizando el protocolo OAuth2
PLAIN	<code>POP3 authentication plain</code>	Autenticación utilizando el protocolo PLAIN

## .checkConnection()

► Histórico

.checkConnection() : Object

Parámetros	Tipo		Descripción
Resultado	Objeto	<-	Estado de la conexión del objeto transportador

### Descripción

La función `.checkConnection()` comprueba la conexión utilizando la información almacenada en el objeto transporter, recrea la conexión si es necesario y devuelve el estado. Esta función permite verificar que los valores proporcionados por el usuario son válidos y coherentes.

### Objeto devuelto

La función envía una solicitud al servidor de correo y devuelve un objeto que describe el estado del correo. Este objeto puede contener las siguientes propiedades:

Propiedad		Tipo	Descripción
success		booleano	True si la verificación es exitosa, False en caso contrario
status		number	(sólo SMTP) Código de estado devuelto por el servidor de correo (0 en caso de un problema no relacionado con el procesamiento del correo)
statusText		texto	Mensaje de estado devuelto por el servidor de correo, o último error devuelto en la pila de errores de 4D
errors		colección	Pila de errores 4D (no se devuelve si se recibe una respuesta del servidor de correo)
	[ ].errCode	number	Código de error 4D
	[ ].message	texto	Descripción del error 4D
	[ ].componentSignature	texto	Firma del componente interno que ha devuelto el error

## Ejemplo

```
var $pw : Text
var $options : Object
$options:=New object

$pw:=Request("Please enter your password:")
if(OK=1)
    $options.host:="pop3.gmail.com"

$options.user:="test@gmail.com"
$options.password:=$pw

$transporter:=POP3 New transporter($options)

$status:=$transporter.checkConnection()
If($status.success)
    ALERT("POP3 connection check successful!")
Else
    ALERT("Error: "+$status.statusText)
End if
End if
```

## .connectionTimeOut

► Histórico

.connectionTimeOut : Integer

### Descripción

La propiedad `.connectionTimeOut` contiene el tiempo máximo de espera (en segundos) permitido para establecer una conexión con el servidor. Por defecto, si la propiedad no se ha definido en el objeto servidor (utilizado para crear el objeto transportador con `SMTP New transporter`, `POP3 New transporter`, o `IMAP New transporter`), el valor es 30.

## .delete()

► Histórico

.delete( *msgNumber* : Integer )

Parámetros	Tipo		Descripción
<i>msgNumber</i>	Integer	->	Número del mensaje a eliminar

### Descripción

La función `.delete( )` marca el email *msgNumber* a eliminar en el servidor POP3.

En el parámetro *msgNumber*, pase el número del correo electrónico a eliminar. Este número es devuelto en la propiedad `number` por el método `.getMailInfoList()`.

La ejecución de este método no elimina realmente ningún correo electrónico. El correo electrónico marcado se eliminará del servidor POP3 sólo cuando se destruya el objeto `POP3_transporter` (creado con `POP3 New transporter`). El marcador también puede eliminarse utilizando el método `.undeleteAll()`.

Si la sesión actual termina inesperadamente y se cierra la conexión (por ejemplo, por tiempo de espera, fallo de la red, etc.), se genera un mensaje de error y los mensajes marcados para ser borrados permanecerán en el servidor POP3.

## Ejemplo

```

$mailInfoList:=$POP3_transporter.getMailInfoList()
For each($mailInfo;$mailInfoList)
    // Marcar el correo como "a eliminar al final de la sesión"
    $POP3_transporter.delete($mailInfo.number)
End for each
    // Forzar el cierre de sesión para eliminar los correos marcados para ser borrados
CONFIRM("Selected messages will be deleted.;";"Delete;";"Undo")
If(OK=1) //borrado confirmado
    $POP3_transporter:=Null
Else
    $POP3_transporter.undeleteAll() //eliminar los marcadores de eliminación
End if

```

## .getBoxInfo()

► Histórico

.getBoxInfo() : Object

Parámetros	Tipo		Descripción
Resultado	Objeto	<-	objeto boxInfo

Descripción

La función `.getBoxInfo()` devuelve un objeto `boxInfo` correspondiente al buzón designado por el [Transporter](#). Esta función permite recuperar la información sobre el buzón.

El objeto `boxInfo` devuelto contiene las siguientes propiedades:

Propiedad	Tipo	Descripción
mailCount	Número	Número de mensajes en el buzón
size	Número	Tamaño del mensaje en bytes

Ejemplo

```

var $server; $boxinfo : Object

$server:=New object
$server.host:="pop.gmail.com" //Obligatorio
$server.port:=995
$server.user:="4d@gmail.com"
$server.password:="XXXXXXXX"

$transporter:=POP3 New transporter($server)

//mailbox info
$boxInfo:=$transporter.getBoxInfo()
ALERT("The mailbox contains "+String($boxInfo.mailCount)+" messages.")

```

## .getMail()

► Histórico

.getMail( *msgNumber* : Integer ) : Object

Parámetros	Tipo		Descripción
msgNumber	Integer	->	Número del mensaje en la lista
Resultado	Objeto	<-	Objeto Email

## Descripción

La función `.getMailInfo()` devuelve un objeto `mailInfo` correspondiente al `msgNumber` en el buzón designado por el `POP3 transporter`. Esta función permite recuperar la información sobre el email.

Pase en `msgNumber` el número del mensaje a recuperar. Este número es devuelto en la propiedad `number` por la función `.getMailInfoList()`.

El método devuelve Null si:

- `msgNumber` designa un mensaje inexistente,
- el mensaje se marcó para su eliminación utilizando `.delete()`.

## Objeto devuelto

`.getMail()` devuelve un objeto `Email`.

## Ejemplo

Quiere saber el remitente del primer correo del buzón:

```
var $server; $transporter : Object
var $mailInfo : Collection
var $sender : Variant

$server:=New object
$server.host:="pop.gmail.com" //Obligatorio
$server.port:=995
$server.user:="4d@gmail.com"
$server.password:="XXXXXXXX"

$transporter:=POP3 New transporter($server)

$mailInfo:=$transporter.getMailInfoList()
$sender:=$transporter.getMail($mailInfo[0].number).from
```

## `.getMailInfo()`

► Histórico

`getMailInfo( msgNumber : Integer ) : Object`

Parámetros	Tipo		Descripción
msgNumber	Integer	->	Número del mensaje en la lista
Resultado	Objeto	<-	objeto mailInfo

## Descripción

La función `.getMailInfoList()` devuelve una colección de objetos `mailInfo` que describen todos los mensajes del buzón designado por el `POP3 transporter`. Esta función permite gestionar localmente la lista de mensajes localizados en el servidor de correo POP3.

En `msgNumber`, pase el número del mensaje a recuperar. Este número es devuelto en la propiedad `number` por el método `.getMailInfoList()`.

El objeto `mailInfo` devuelto contiene las siguientes propiedades:

Propiedad	Tipo	Descripción
size	Número	Tamaño del mensaje en bytes
id	Texto	ID único del mensaje

El método devuelve Null si:

- *msgNumber* designa un mensaje inexistente,
- el mensaje se marcó para su eliminación utilizando `.delete()`.

#### Ejemplo

```
var $server; $mailInfo : Object
var $mailNumber : Integer

$server.host:="pop.gmail.com" //Obligatorio
$server.port:=995
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

var $transporter : 4D.POP3Transporter
$transporter:=POP3 New transporter($server)

//message info
$mailInfo:=$transporter.getMailInfo(1) //obtener el primer e-mail
If($mailInfo #Null)
  ALERT("First mail size is:"+String($mailInfo.size)+" bytes.")
End if
```

## .getMailInfoList()

► Histórico

.getMailInfoList() : Collection

Parámetros	Tipo		Descripción
Resultado	Collection	<-	Colección de objetos <code>mailInfo</code>

#### Descripción

Summary -->configura una nueva conexión POP3en función del parámetro *servidor* y devuelve un nuevo objeto [POP3 transporter](#). El objeto transportador devuelto se utilizará normalmente para recibir correos electrónicos. Esta función permite gestionar localmente la lista de mensajes localizados en el servidor de correo POP3.

Cada objeto `mailInfo` de la colección devuelta contiene las siguientes propiedades:

Propiedad	Tipo	Descripción
[ ].size	Número	Tamaño del mensaje en bytes
[ ].number	Número	Número del mensaje
[ ].id	Texto	ID único del mensaje (útil si almacena el mensaje localmente)

Si el buzón no contiene ningún mensaje, se devuelve una colección vacía.

#### Propiedades number e ID

*number* es el número de un mensaje del buzón en el momento en que se creó el `POP3_transporter`. La propiedad *number* no es un valor estático en relación con ningún mensaje específico y cambiará de una sesión a otra dependiendo de su relación con otros mensajes en el buzón en el momento en que se abrió la sesión. Los números asignados a los mensajes sólo son válidos durante la vigencia del `POP3_transporter`. En el momento en que el `POP3_transporter` sea

eliminado cualquier mensaje marcado para ser borrado será eliminado. Cuando el usuario vuelve a conectarse al servidor, los mensajes actuales en el buzón serán reenumerados de 1 a x.

Sin embargo, el *id* es un número único asignado al mensaje cuando fue recibido por el servidor. Este número se calcula utilizando la hora y la fecha de recepción del mensaje y es un valor asignado por su servidor POP3. Lamentablemente, los servidores POP3 no utilizan el *id* como referencia principal de sus mensajes. A lo largo de las sesiones POP3 deberá especificar el *number* como referencia a los mensajes del servidor. Los desarrolladores deben tener cierto cuidado si desarrollan soluciones que introducen referencias a los mensajes en una base de datos pero dejan el cuerpo del mensaje en el servidor.

## Ejemplo

Quiere saber el número total y el tamaño de los correos electrónicos en el buzón:

```
var $server : Object
$server:=New object
$server.host:="pop.gmail.com" //Obligatorio
$server.port:=995
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

var $transporter : 4D.POP3Transporter
$transporter:=POP3 New transporter($server)

C_COLLECTION($mailInfo)
C_LONGINT($vNum;$vSize)

$mailInfo:=$transporter.getMailInfoList()
$vNum:=$mailInfo.length
$vSize:=$mailInfo.sum("size")

ALERT("The mailbox contains "+String($vNum)+" message(s) for "+String($vSize)+" bytes.")
```

## .getMIMEAsBlob()

► Histórico

.getMIMEAsBlob( *msgNumber* : Integer ) : Blob

Parámetros	Tipo		Descripción
<i>msgNumber</i>	Integer	->	Número del mensaje en la lista
Resultado	Blob	<-	Blob de la cadena MIME devuelta por el servidor de correo

### Descripción

La función `.getMIMEAsBlob()` devuelve un BLOB con el contenido MIME del mensaje correspondiente al *msgNumber* en el buzón designado por el `POP3_transporter`.

En *msgNumber*, pase el número del mensaje a recuperar. Este número es devuelto en la propiedad *number* por el método `.getMailInfoList()`.

El método devuelve un BLOB vacío si:

- *msgNumber* designa un mensaje inexistente,
- el mensaje se marcó para su eliminación utilizando `.delete()`.

### BLOB devuelto

`.getMIMEAsBlob()` devuelve un BLOB que puede almacenarse en una base de datos o convertirse en un objeto `Email` con el comando `MAIL Convert from MIME`.

## Ejemplo

Quiere saber el número total y el tamaño de los correos electrónicos en el buzón:

```
var $server : Object
var $mailInfo : Collection
var $blob : Blob
var $transporter : 4D.POP3Transporter

$server:=New object
$server.host:="pop.gmail.com"
$server.port:=995
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

$transporter:=POP3 New transporter($server)

$mailInfo:=$transporter.getMailInfoList()
$blob:=$transporter.getMIMEAsBlob($mailInfo[0].number)
```

## .host

► Histórico

.host : Text

### Descripción

La propiedad `.host` contiene el nombre o la dirección IP del servidor local. Se utiliza para las transacciones de correo (SMTP, POP3, IMAP).

## .logFile

► Histórico

.logFile : Text

### Descripción

La propiedad `.logFile` contiene la ruta del archivo de registro extendido definido (si existe) para la conexión de correo. Puede ser relativo (a la carpeta actual Logs) o absoluto.

A diferencia de los archivos de registro clásicos (habilitados mediante el comando `SET DATABASE PARAMETER`), los archivos de registro extendidos almacenan el contenido MIME de todos los correos enviados y no tienen ningún límite de tamaño. Para más información sobre los archivos de registro extendidos, consulte:

- Conexiones SMTP - [4DSMTPLLog.txt](#)
- \*\*Conexiones POP3 \*\* - [4DSMTPLLog.txt](#)
- Conexiones IMAP - [4DIMAPLog.txt](#)

## .port

► Histórico

.port : Integer

### Descripción

La propiedad `.port` contiene el número de puerto utilizado para las transacciones de correo. Por defecto, si la propiedad `port` no se ha definido en el objeto `server` (utilizado para crear el objeto transportador con `SMTP New transporter`, `POP3 New transporter`, `IMAP New transporter`), el puerto utilizado es:

- SMTP - 587
- POP3 - 995

- IMAP - 993

## .undeleteAll()

► Histórico

.undeleteAll()

Parámetros	Tipo	Descripción	-----	----	::	-----		No requiere ningún parámetro
------------	------	-------------	-------	------	----	-------	--	------------------------------

### Descripción

La función `.undeleteAll()` elimina todas las banderas de borrado definidas en los correos electrónicos en el [POP3\\_transporter](#).

## .user

► Histórico

.user : Text

### Descripción

La propiedad `.user` contiene el nombre de usuario utilizado para la autenticación en el servidor de correo.

# Sesión

Los objetos Session son devueltos por el comando `Session` cuando [se habilitan las sesiones escalables en su proyecto](#). El objeto Session es creado y mantenido automáticamente por el servidor web 4D para controlar la sesión de un cliente web (por ejemplo, un navegador). Este objeto proporciona al desarrollador web una interfaz para la sesión de usuario, permitiendo gestionar privilegios, almacenar datos contextuales, compartir información entre procesos y lanzar procesos preventivos relacionados con la sesión.

Para obtener información detallada sobre la implementación de la sesión, consulte la sección [Sesiones del servidor web](#).

## Resumen

<code>.clearPrivileges()</code>	elimina todos los privilegios asociados a la sesión
<code>.expirationDate : Text</code>	la fecha y hora de expiración de la cookie de sesión
<code>.hasPrivilege( privilege : Text ) : Boolean</code>	devuelve True si el privilegio está asociado a la sesión, y False en caso contrario
<code>.idleTimeout : Integer</code>	el tiempo de inactividad de la sesión (en minutos), después del cual la sesión es cerrada automáticamente por 4D
<code>.isGuest() : Boolean</code>	devuelve True si la sesión es una sesión Guest (es decir, no tiene privilegios)
<code>.setPrivileges( privilege : Text )</code> <code>.setPrivileges( privileges : Collection )</code> <code>.setPrivileges( settings : Object )</code>	asocia los privilegios definidos en el parámetro a la sesión
<code>.storage : Object</code>	un objeto compartido que puede ser utilizado para almacenar información disponible para todas las peticiones del cliente web
<code>.userName : Text</code>	el nombre de usuario asociado a la sesión

# Sesión

► Histórico

Session : 4D.Session

Parámetros	Tipo		Descripción
Resultado	4D.Session	<-	Objeto Session

## Descripción

El comando `Session` devuelve el objeto `Session` correspondiente a la sesión web actual del usuario escalable.

Este comando sólo funciona cuando [están activadas las sesiones escalables](#). Devuelve `Null` cuando las sesiones están deshabilitadas o cuando se utilizan sesiones heredadas.

Cuando se habilitan las sesiones escalables, el objeto `Session` está disponible desde cualquier proceso web en los siguientes contextos:

- Métodos base `On Web Authentication`, `On Web Connection`, y `On REST Authentication`,
- Las [funciones Data Model Class](#) ORDA llamadas por las peticiones REST,
- código procesado a través de las etiquetas 4D en las páginas semidinámicas (4DTEXT, 4DHML, 4DEVAL, 4DSCRIPT/, 4DCODE)
- los métodos proyecto con el atributo "Available through 4D tags and URLs (4DACTION...)" y llamados a través de 4DACTION/ urls.

## Ejemplo

Ha definido el método `action_Session` con el atributo "Available through 4D tags and URLs". Se llama al método introduciendo la siguiente URL en el navegador:

IP:port/4DACTION/action\_Session

```
//método action_Session
Case of
:(Session#Null)
  If(Session.hasPrivilege("WebAdmin")) //llamada de la función hasPrivilege
    WEB SEND TEXT("4DACTION --> Session is WebAdmin")
  Else
    WEB SEND TEXT("4DACTION --> Session is not WebAdmin")
  End if
Else
  WEB SEND TEXT("4DACTION --> Session is null")
End case
```

## .clearPrivileges()

► Histórico

.clearPrivileges() | Parámetros | Tipo | Descripción | | ----- | ---- |::| ----- | | | | No requiere ningún parámetro |

### Descripción

La función `.clearPrivileges()` elimina todos los privilegios asociados a la sesión. Como resultado, la sesión se convierte automáticamente en una sesión de invitado.

## Ejemplo

```
//Iniciar una sesión
var $isGuest : Boolean

Session.clearPrivileges()
$isGuest:=Session.isGuest() //isGuest es True
```

## .expirationDate

► Histórico

.expirationDate : Text

### Descripción

La propiedad `.expirationDate` contiene la fecha y hora de expiración de la cookie de sesión. El valor se expresa como

texto en el formato ISO 8601: `YYYY-MM-DDTHH:MM:SS.mmmZ`.

Esta propiedad es de sólo lectura. Se vuelve a calcular automáticamente si se modifica el valor de la propiedad `.idleTimeout`.

## Ejemplo

```
var $expiration : Text  
$expiration:=Session.expirationDate //eg "2021-11-05T17:10:42Z"
```

## .hasPrivilege()

► Histórico

`.hasPrivilege( privilege : Text ) : Boolean`

Parámetros	Tipo		Descripción
privilege	Texto	<-	Nombre del privilegio a verificar
Resultado	Booleano	<-	True si la sesión tiene <i>privilege</i> , False en caso contrario

### Descripción

La función `.hasPrivilege()` devuelve True si el privilegio está asociado a la sesión, y False en caso contrario.

## Ejemplo

Quiere comprobar si el privilegio "WebAdmin" está asociado a la sesión:

```
If (Session.hasPrivilege("WebAdmin"))  
    //El acceso está concedido, no haga nada  
Else  
    //Mostrar una página de autenticación  
  
End if
```

## .idleTimeout

► Histórico

`.idleTimeout : Integer`

### Descripción

La propiedad `.idleTimeout` contiene el tiempo de inactividad de la sesión (en minutos), después del cual la sesión es cerrada automáticamente por 4D.

Si no se define esta propiedad, el valor por defecto es 60 (1h).

Cuando se define esta propiedad, la propiedad `.expirationDate` se actualiza en consecuencia.

El valor no puede ser inferior a 60: si se define un valor inferior, el tiempo de espera se eleva hasta 60.

Esta propiedad es de sólo escritura.

## Ejemplo

```

If (Session.isGuest())
    // Una sesión de invitado se cerrará tras 60 minutos de inactividad
    Session.idleTimeout:=60
Else
    // Las demás sesiones se cerrarán tras 120 minutos de inactividad
    Session.idleTimeout:=120
End if

```

## .isGuest()

► Histórico

.isGuest() : Boolean

Parámetros	Tipo		Descripción
Resultado	Booleano	<-	True si la sesión es una sesión Guest, False en caso contrario

### Descripción

La función `.isGuest()` devuelve True si la sesión es una sesión Guest (es decir, no tiene privilegios).

### Ejemplo

En el método base `On Web Connection`:

```

If (Session.isGuest())
    //Hacer algo para el usuario invitado
End if

```

## .setPrivileges()

► Histórico

.setPrivileges( *privilege* : Text )  
.setPrivileges( *privileges* : Collection )  
.setPrivileges( *settings* : Object )

Parámetros	Tipo		Descripción
<i>privilege</i>	Texto	->	Nombre del privilegio
<i>privileges</i>	Collection	->	Colección de nombres de privilegios
<i>parámetros</i>	Objeto	->	Objeto con una propiedad "privilegios" (cadena o colección)

### Descripción

La función `.setPrivileges()` asocia los privilegios definidos en el parámetro a la sesión.

- En el parámetro *privilege*, pase una cadena que contenga un nombre de privilegio (o varios nombres de privilegio separados por comas).
- En el parámetro *privileges*, pase una colección de cadenas que contengan nombres de privilegios.
- En el parámetro *settings*, pase un objeto que contenga las siguientes propiedades:

Propiedad	Tipo	Descripción
privileges	Text o Collection	<ul style="list-style-type: none"> <li>Cadena que contiene un nombre de privilegio, o</li> <li>Colección de cadenas que contienen nombres de privilegios</li> </ul>
userName	Texto	Nombre de usuario para asociar a la sesión (opcional)

Si la propiedad `privileges` contiene un nombre de privilegio no válido, se ignora.

En la implementación actual, sólo está disponible el privilegio "WebAdmin".

Por defecto, cuando no se asocia ningún privilegio a la sesión, ésta es una [sesión invitado](#).

La propiedad `userName` está disponible a nivel de objeto de sesión (sólo lectura).

## Ejemplo

En un método de autenticación personalizado, se establece el privilegio "WebAdmin" para el usuario:

```
var $userOK : Boolean
...
... //Autenticar al usuario
If ($userOK) //El usuario ha sido aprobado
  var $info : Object
  $info:=New object()
  $info.privileges:=New collection("WebAdmin")
  Session.setPrivileges($info)
End if
```

## .storage

► Histórico

`.storage` : Object

### Descripción

La propiedad `.storage` contiene un objeto compartido que puede ser utilizado para almacenar información disponible para todas las peticiones del cliente web.

Cuando se crea un objeto `Session`, la propiedad `.storage` está vacía. Al ser un objeto compartido, esta propiedad estará disponible en el objeto `Storage` del servidor.

Al igual que el objeto `Storage` del servidor, la propiedad `.storage` es siempre "única": añadir un objeto compartido o una colección compartida a `.storage` no crea un grupo compartido.

Esta propiedad es sólo lectura en sí misma pero devuelve un objeto de lectura-escritura.

## Ejemplo

Se desea almacenar la IP del cliente en la propiedad `.storage`. Puede escribir en el método de base de datos `On Web Authentication`:

```
If (Session.storage.clientIP=Null) //primer acceso
    Use (Session.storage)
        Session.storage.clientIP:=New shared object("value"; $clientIP)
    End use
End if
```

## .userName

► Histórico

.userName : Text

### Descripción

La propiedad `.userName` contiene el nombre de usuario asociado a la sesión. Puede utilizarlo para identificar al usuario dentro de su código.

Esta propiedad es una cadena vacía por defecto. Se puede establecer mediante la propiedad `privileges` de la función `setPrivileges()`.

Esta propiedad esde sólo lectura.

# Signal

Las señales son herramientas que ofrece el lenguaje 4D para gestionar las interacciones y evitar conflictos entre procesos en una aplicación multiproceso. Las señales le permiten asegurarse de que uno o más procesos esperarán a que se complete una tarea específica antes de continuar la ejecución. Todo proceso puede esperar y/o liberar una señal.

Los semáforos también pueden utilizarse para gestionar las interacciones. Los semáforos permiten asegurarse de que dos o más procesos no modifican el mismo recurso (archivo, registro...) al mismo tiempo. Sólo el proceso que define el semáforo puede eliminarlo.

## Objeto signal

Una señal es un objeto compartido que debe ser pasado como parámetro a los comandos que llaman o crean trabajadores o procesos.

Un objeto `4D.Signal` contiene los siguientes métodos y propiedades integrados:

- `.wait()`
- `.trigger()`
- `.signaled`
- `.description`.

Todo worker/proceso que llame al método `.wait()` suspenderá su ejecución hasta que la propiedad `.signaled` sea verdadera. Mientras espera una señal, el proceso que llama no utiliza ninguna CPU. Esto puede ser muy interesante para el rendimiento en aplicaciones multiproceso. La propiedad `.signaled` se convierte en true cuando cualquier worker/proceso llama al método `.trigger()`.

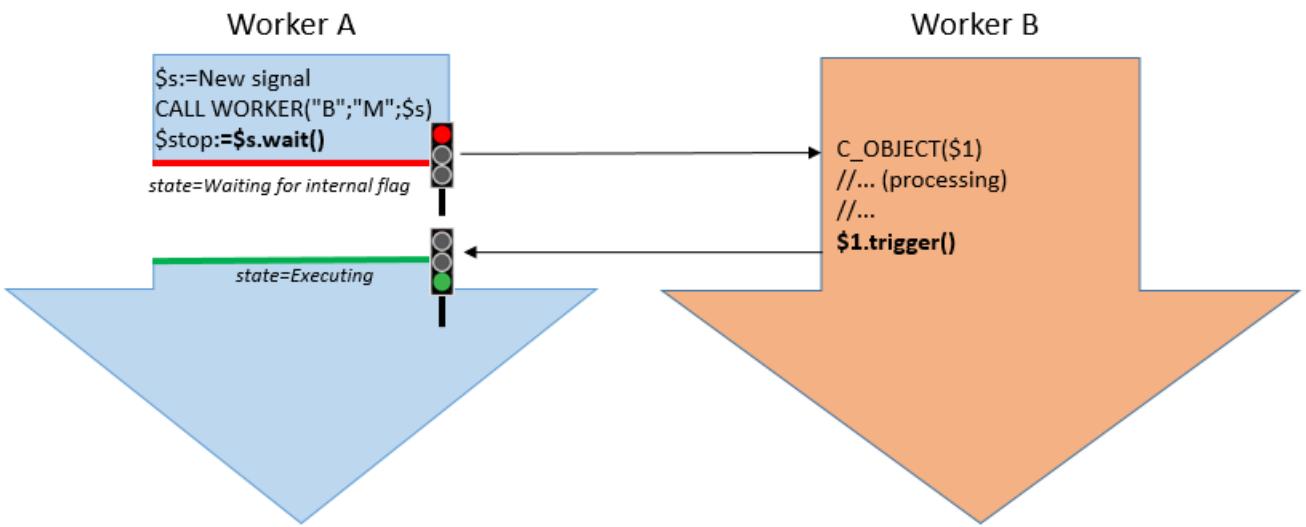
Tenga en cuenta que para evitar situaciones de bloqueo, el `.wait()` también puede regresar después de que se haya alcanzado un tiempo de espera definido.

Los objetos signal se crean con el comando [New signal](#).

## Trabajar con señales

En 4D, se crea un nuevo objeto signal llamando al comando [New signal](#). Una vez creada, esta señal debe pasarse como parámetro a los comandos `New process` o `CALL WORKER` para que la modifiquen cuando hayan terminado la tarea que quiere esperar.

- `signal.wait()` debe ser llamado desde el worker/proceso que necesita que otro worker/proceso termine una tarea para poder continuar.
- `signal.trigger()` debe llamarse desde el worker/proceso que terminó su ejecución para liberar a todos los demás.



Una vez que una señal ha sido liberada utilizando una llamada `signal.trigger()`, no puede ser reutilizada de nuevo. Si desea definir otra señal, debe llamar de nuevo al comando `New signal`.

Dado que un objeto signal es un [objeto compartido](#), puede utilizarlo para devolver resultados de los workers/procesos llamados, siempre que no olvide escribir los valores dentro de una estructura `Use...End use` (ver ejemplo).

## Ejemplo

```

var $signal : 4D.Signal

// Creación de un signal
$signal:=New signal

// llamar al proceso principal y ejecutar el método OpenForm
CALL WORKER(1;"OpenForm";$signal)
// hacer otro cálculo
...
// Esperando el final del proceso
$signaled:=$signal.wait()

// Procesamiento de los resultados
$calc:=$signal.result+...

```

Método `OpenForm`:

```

#DECLARE ($signal : 4D.Signal)
var $form : Object
$form:=New object("value";0)

// Abrir el formulario
$win:=Open form window("Information";Movable form dialog box)
DIALOG("Information";$form)
CLOSE WINDOW($win)

// Añade un nuevo atributo a su objeto compartido $signal para pasar su resultado al otro proceso:
Use($signal)
    $signal.result:=$form.value
End use

// Activar la señal al proceso de espera
$signal.trigger()

```

## Resumen

#### `.description : Text`

contiene una descripción personalizada para el objeto `Signal`.

#### `.signaled: Boolean`

contiene el estado actual del objeto `Signal`

#### `.trigger()`

pone la propiedad `signaled` del objeto signal como true

#### `.wait( { timeout : Real } ) : Boolean`

hace que el proceso actual espere hasta que la propiedad `.signaled` del objeto signal se convierta en true o expire el `timeout` opcional.

Para evitar que el código se bloquee, puede pasar un tiempo máximo de espera en segundos en el parámetro `timeout` (se aceptan decimales).

Atención: la llamada a `.wait( )` sin un `timeout` en el proceso principal de 4D no es recomendable porque podría congelar toda la aplicación 4D.

Si signal ya está en el estado de señalización (es decir, la propiedad `.signaled` ya es true), la función devuelve inmediatamente, sin esperar.

La función devuelve el valor de la propiedad `.signaled`. La evaluación de este valor permite saber si la función devuelta porque el `.trigger( )` se ha llamado (`.signaled` es true) o si el `timeout` vencido (`.signaled` es false).

El estado de un proceso que espera un signal es `Waiting for internal flag`.

## New signal

► Histórico

New signal { ( `description` : Text ) } : 4D.Signal

Parámetros	Tipo		Descripción
<code>description</code>	Texto	->	Descripción para la señal
Resultado	4D.Signal	<-	Objeto nativo que encapsula la señal

### Descripción

El comando `New signal` crea un objeto `4D.Signal`.

Una señal es un objeto compartido que puede ser pasado como parámetro de un worker o proceso a otro worker o proceso, de manera que:

- el worker/proceso llamado puede actualizar el objeto de la señal después de que se haya completado el procesamiento específico
- el worker/proceso que llama puede detener su ejecución y esperar hasta que se actualice la señal, sin consumir recursos de la CPU.

Opcionalmente, en el parámetro `description` puede pasar un texto personalizado que describa la señal. Este texto también puede definirse después de la creación de la señal.

Dado que el objeto señal es un objeto compartido, también se puede utilizar para mantener las propiedades del usuario, incluyendo la propiedad `.description`, llamando a la estructura `Use...End use`.

Valor devuelto

Un nuevo objeto `4D.Signal`.

## Ejemplo

Este es un ejemplo típico de un worker que fija una señal:

```
var $signal : 4D.Signal  
$signal:=New signal("This is my first signal")  
  
CALL WORKER("myworker";"doSomething";$signal)  
$signaled:=$signal.wait(1) //espera 1 segundo como máximo  
  
If($signaled)  
    ALERT("myworker finished the work. Result: "+$signal.myresult)  
Else  
    ALERT("myworker no ha terminado en menos de 1s")  
End if
```

El método `doSomething` puede ser:

```
#DECLARE ($signal : 4D.Signal)  
//todo proceso  
//...  
Use($signal)  
    $signal.myresult:=$processingResult //devolver el resultado  
End use  
$signal.trigger() //La operación se ha terminado
```

## .description

► Histórico

`.description` : Text

### Descripción

La propiedad `.description` contiene una descripción personalizada para el objeto `Signal` ..

`.description` puede definirse al crear el objeto `signal` o en cualquier momento. Tenga en cuenta que, dado que el objeto `Signal` es un objeto compartido, cualquier acceso en modo de escritura a la propiedad `.description` debe estar rodeado por una estructura `Use...End use` .

Esta propiedad es lectura-escritura.

## .signaled

► Histórico

`.signaled`: Boolean

### Descripción

La propiedad `.signaled` contiene el estado actual del objeto `Signal` . Cuando se crea `signal`, `.signaled` es `False`. Se convierte en `True` cuando se llama al objeto `.trigger()` .

Esta propiedad es de sólo lectura.

## .trigger()

► Histórico

## .trigger( )

Parámetros	Tipo	Descripción	-----	----	::  -----		No requiere ningún parámetro
------------	------	-------------	-------	------	-----------	--	------------------------------

### Descripción

La función `.trigger( )` pone la propiedad `signaled` del objeto `signal` como `true` y despierta a todos los workers o procesos que esperan esta señal.

Si la señal ya está en el estado de señalización (es decir, la propiedad `signaled` ya es `true`), la función no hace nada.

## .wait()

► Histórico

## .wait( { *timeout* : Real } ) : Boolean

Parámetros	Tipo		Descripción
<code>timeout</code>	Real	->	Tiempo máximo de espera de la señal en segundos
Resultado	Booleano	<-	Estado de la propiedad <code>.signaled</code>

### Descripción

La función `.wait( )` hace que el proceso actual espere hasta que la propiedad `.signaled` del objeto `signal` se convierta en `true` o expire el `timeout` opcional.

Para evitar que el código se bloquee, puede pasar un tiempo máximo de espera en segundos en el parámetro `timeout` (se aceptan decimales).

Atención: la llamada a `.wait( )` sin un `timeout` en el proceso principal de 4D no es recomendable porque podría congelar toda la aplicación 4D.

Si `signal` ya está en el estado de señalización (es decir, la propiedad `.signaled` ya es `true`), la función devuelve inmediatamente, sin esperar.

La función devuelve el valor de la propiedad `.signaled`. La evaluación de este valor permite saber si la función devuelta porque el `.trigger( )` se ha llamado (`.signaled` es `true`) o si el `timeout` vencido (`.signaled` es `false`).

El estado de un proceso que espera un signal es `Waiting for internal flag`.

# SMTPTransporter

La clase `SMTPTransporter` permite configurar conexiones SMTP y enviar correos electrónicos a través de objetos `SMTP transporter`.

## Objeto SMTP Transporter

Los objetos SMTP Transporter se instancian con el comando `SMTP New transporter`. Ofrecen las siguientes propiedades y funciones:

<code>.acceptUnsecureConnection: Boolean</code>	True si se permite a 4D establecer una conexión no cifrada
<code>.authenticationMode: Text</code>	el modo de autenticación utilizado para abrir la sesión en el servidor de correo
<code>.bodyCharset : Text</code>	el conjunto de caracteres y la codificación utilizados para la parte del cuerpo del correo electrónico
<code>.checkConnection() : Object</code>	comprueba la conexión utilizando la información almacenada en el objeto transporter
<code>.connectionTimeOut : Integer</code>	el tiempo máximo de espera (en segundos) permitido para establecer una conexión con el servidor
<code>.headerCharset : Text</code>	el conjunto de caracteres y la codificación utilizados para el encabezado del correo electrónico
<code>.host : Text</code>	el nombre o la dirección IP del servidor local
<code>.keepAlive : Boolean</code>	True si la conexión SMTP debe permanecer activa hasta que el objeto sea destruido <code>transporter</code>
<code>.logFile : Text</code>	la ruta del archivo de registro extendido definido (si existe) para la conexión de correo
<code>.port : Integer</code>	el número de puerto utilizado para las transacciones de correo
<code>.send( mail : Object ) : Object</code>	envía el objeto <code>mail object</code> al servidor SMTP definido en el objeto <code>transporter</code> y devuelve un objeto estado
<code>.sendTimeOut : Integer</code>	el tiempo máximo de espera (en segundos) de una llamada a <code>.send()</code> antes de que se produzca un timeout
<code>.user : Text</code>	el nombre de usuario utilizado para la autenticación en el servidor de correo

## SMTP New transporter

► Histórico

`SMTP New transporter( server : Object ) : 4D.SMTPTransporter`

Parámetros	Tipo		Descripción
server	Objeto	->	Información del servidor de correo
Resultado	4D.SMTPTransporter	<-	Objeto SMTP transporter

## Descripción

El comando `SMTP New transporter` configura una nueva conexión SMTP en función del parámetro `server` y devuelve un nuevo objeto `SMTP transporter`. El objeto transportador devuelto se utilizará normalmente para el envío de correos electrónicos.

Este comando no abre ninguna conexión con el servidor SMTP. La conexión SMTP se abre realmente cuando la función `.send()` se ejecuta.

La conexión SMTP se cierra automáticamente \* cuando se destruye el objeto transportador si la propiedad `keepAlive` es true (por defecto), \* después de cada ejecución de la función `.send( )` si la propiedad `keepAlive` está en false.

En el parámetro `server`, pase un objeto que contenga las siguientes propiedades:

<code>server</code>		Valor por defecto (si se omite)
<code>.acceptUnsecureConnection: Boolean</code> True si se permite a 4D establecer una conexión no cifrada		False
<code>.accessTokenOAuth2: Text</code> <code>.accessTokenOAuth2: Object</code> Cadena de texto u objeto token que representan las credenciales de autorización OAuth 2. Sólo se utiliza con <code>OAUTH2 authenticationMode</code> . Si se utiliza <code>accessTokenOAuth2</code> pero se omite <code>authenticationMode</code> , se utiliza el protocolo OAuth 2 (si el servidor lo permite). No se devuelve en el objeto <code>SMTP transporter</code> .		ninguno
<code>.authenticationMode: Text</code> el modo de autenticación utilizado para abrir la sesión en el servidor de correo		se utiliza el modo de autenticación más seguro soportado por el servidor
<code>.bodyCharset : Text</code> el conjunto de caracteres y la codificación utilizados para la parte del cuerpo del correo electrónico		<code>mail mode UTF8 (US-ASCII_UTF8_QP)</code>
<code>.connectionTimeOut : Integer</code> el tiempo máximo de espera (en segundos) permitido para establecer una conexión con el servidor		30
<code>.headerCharset : Text</code> el conjunto de caracteres y la codificación utilizados para el encabezado del correo electrónico		<code>mail mode UTF8 (US-ASCII_UTF8_QP)</code>
<code>.host : Text</code> el nombre o la dirección IP del servidor local		<i>obligatorio</i>
<code>.keepAlive : Boolean</code> True si la conexión SMTP debe permanecer activa hasta que el objeto sea destruido <code>transporter</code>		True
<code>.logFile : Text</code> la ruta del archivo de registro extendido definido (si existe) para la conexión de correo		ninguno
<code>password : Text</code> Contraseña del usuario para la autenticación en el servidor. No se devuelve en el objeto <code>SMTP transporter</code> .		ninguno
<code>.port : Integer</code> el número de puerto utilizado para las transacciones de correo		587
<code>.sendTimeOut : Integer</code> el tiempo máximo de espera (en segundos) de una llamada a <code>.send( )</code> antes de que se produzca un timeout		100
<code>.user : Text</code> el nombre de usuario utilizado para la autenticación en el servidor de correo		ninguno

## Resultado

La función devuelve un `SMTP transporter`. Todas las propiedades devueltas son de sólo lectura.

## Ejemplo

```
$server:=New object
$server.host:="smtp.gmail.com" //Obligatorio
$server.port:=465
$server.user:="4D@gmail.com"
$server.password:="XXXX"
$server.logFile:="LogTest.txt" //Log extendido a guardar en la carpeta Logs

var $transporter : 4D.SMTPTransporter
$transporter:=SMTP New transporter($server)

$email:=New object
$email.subject:="my first mail "
$email.from:="4d@gmail.com"
$email.to:="4d@4d.com;test@4d.com"
$email.textBody:="Hello World"
$email.htmlBody:="

# Hello World



#### 'Neque porro quisquam est qui dolorem ipsum quia dolor sit am <p>There are many variations of passages of Lorem Ipsum available."\ +"The generated Lorem Ipsum is therefore always free from repetition, injected humour, or non-character $status:=$transporter.send($email) If(Not($status.success)) ALERT("An error occurred sending the mail: "+$status.message) End if


```

## 4D.SMTPTransporter.new()

4D.SMTPTransporter.new( server : Object ) : 4D.SMTPTransporter

Parámetros	Tipo		Descripción
server	Objeto	->	Información del servidor de correo
Resultado	4D.SMTPTransporter	<-	Objeto SMTP transporter

### Descripción

La función `4D.SMTPTransporter.new()` crea y devuelve un nuevo objeto de tipo `4D.SMTPTransporter` type. Es idéntico al comando `SMTP New transporter` (acceso directo).

## .acceptUnsecureConnection

► Histórico

.acceptUnsecureConnection: Boolean

### Descripción

La propiedad `.acceptUnsecureConnection` contiene True si se permite a 4D establecer una conexión no cifrada cuando la conexión cifrada no es posible.

Contiene False si no se permiten las conexiones no cifradas, en cuyo caso se devuelve un error cuando no es posible la conexión cifrada.

Los puertos seguros disponibles son:

- SMTP
  - 465: SMTPTS
  - 587 o 25: SMTP con actualización STARTTLS si lo soporta el servidor.

- IMAP
  - 143: Puerto IMAP no encriptado
  - 993: IMAP con actualización STARTTLS si lo soporta el servidor
- POP3
  - 110: Puerto POP3 no encriptado
  - 995: POP3 con actualización STARTTLS si lo soporta el servidor.

## .authenticationMode

► Histórico

.authenticationMode: Text

### Descripción

La propiedad `.authenticationMode` contiene el modo de autenticación utilizado para abrir la sesión en el servidor de correo.

Por defecto, se utiliza el modo más seguro soportado por el servidor.

Los valores posibles son:

Valor	Constantes	Comentario
CRAM-MD5	<code>SMTP authentication CRAM MD5</code>	Autenticación utilizando el protocolo CRAM-MD5
LOGIN	<code>SMTP authentication login</code>	Autenticación utilizando el protocolo LOGIN
OAuth2	<code>SMTP authentication OAuth2</code>	Autenticación utilizando el protocolo OAuth2
PLAIN	<code>SMTP authentication plain</code>	Autenticación utilizando el protocolo PLAIN

## .bodyCharset

► Histórico

.bodyCharset : Text

### Descripción

La propiedad `.bodyCharset` contiene el conjunto de caracteres y la codificación utilizados para la parte del cuerpo del correo electrónico.

- subject,
- attachment filename(s),
- nombre del correo electrónico.

Valores posibles:

Constante	Valor	Comentario
mail mode ISO2022JP	US-ASCII_ISO-2022-JP_UTF8_QP	<ul style="list-style-type: none"> <li><i>headerCharset</i>: US-ASCII si es posible, japonés (ISO-2022-JP) &amp; Quoted-printable si es posible, de lo contrario UTF-8 &amp; Quoted-printable</li> <li><i>bodyCharset</i>: US-ASCII si es posible, japonés (ISO-2022-JP) y 7 bits si es posible, de lo contrario UTF-8 &amp; Quoted-printable</li> </ul>
mail mode ISO88591	ISO-8859-1	<ul style="list-style-type: none"> <li><i>headerCharset</i>: ISO-8859-1 &amp; Quoted-printable</li> <li><i>bodyCharset</i>: ISO-8859-1 y 8 bits</li> </ul>
mail mode UTF8	US-ASCII_UTF8_QP	<i>headerCharset</i> & <i>bodyCharset</i> : US-ASCII si es posible, de lo contrario UTF-8 & Quoted-printable (valor por defecto)
mail mode UTF8 in base64	US-ASCII_UTF8_B64	<i>headerCharset</i> & <i>bodyCharset</i> : US-ASCII si es posible, de lo contrario UTF-8 & base64

## .checkConnection()

► Histórico

.checkConnection() : Object

Parámetros	Tipo		Descripción
Resultado	Objeto	<-	Estado de la conexión del objeto transportador

### Descripción

La función `.checkConnection()` comprueba la conexión utilizando la información almacenada en el objeto transporter, recrea la conexión si es necesario y devuelve el estado. Esta función permite verificar que los valores proporcionados por el usuario son válidos y coherentes.

### Objeto devuelto

La función envía una solicitud al servidor de correo y devuelve un objeto que describe el estado del correo. Este objeto puede contener las siguientes propiedades:

Propiedad		Tipo	Descripción
success		booleano	True si la verificación es exitosa, False en caso contrario
status		number	(sólo SMTP) Código de estado devuelto por el servidor de correo (0 en caso de un problema no relacionado con el procesamiento del correo)
statusText		texto	Mensaje de estado devuelto por el servidor de correo, o último error devuelto en la pila de errores de 4D
errors		colección	Pila de errores 4D (no se devuelve si se recibe una respuesta del servidor de correo)
	[ ].errCode	number	Código de error 4D
	[ ].message	texto	Descripción del error 4D
	[ ].componentSignature	texto	Firma del componente interno que ha devuelto el error

Para obtener información sobre los códigos de estado SMTP, consulte [esta página](#).

### Ejemplo

```

var $pw : Text
var $options : Object
var $transporter : 4D.SMTPTransporter
$options:=New object

$pw:=Request("Please enter your password:")
$options.host:="smtp.gmail.com"

$options.user:="test@gmail.com"
$options.password:=$pw

$transporter:=SMTP New transporter($options)

$status:=$transporter.checkConnection()
If($status.success=True)
    ALERT("SMTP connection check successful!")
Else
    ALERT("Error # "+String($status.status)+", "+$status.statusText)
End if

```

## .connectionTimeOut

► Histórico

.connectionTimeOut : Integer

### Descripción

La propiedad `.connectionTimeOut` contiene el tiempo máximo de espera (en segundos) permitido para establecer una conexión con el servidor. Por defecto, si la propiedad no se ha definido en el objeto servidor (utilizado para crear el objeto transportador con `SMTP New transporter`, `POP3 New transporter`, o `IMAP New transporter`), el valor es 30.

## .headerCharset

► Histórico

.headerCharset : Text

### Descripción

La propiedad `.headerCharset` contiene el conjunto de caracteres y la codificación utilizados para el encabezado del correo electrónico. El encabezado incluye las siguientes partes del correo electrónico:

- subject,
- attachment filename(s),
- nombre del correo electrónico.

Valores posibles:

Constante	Valor	Comentario
mail mode ISO2022JP	US-ASCII_ISO-2022-JP_UTF8_QP	<ul style="list-style-type: none"> <li>• <i>headerCharset</i>: US-ASCII si es posible, japonés (ISO-2022-JP) &amp; Quoted-printable si es posible, de lo contrario UTF-8 &amp; Quoted-printable</li> <li>• <i>bodyCharset</i>: US-ASCII si es posible, japonés (ISO-2022-JP) y 7 bits si es posible, de lo contrario UTF-8 &amp; Quoted-printable</li> </ul>
mail mode ISO88591	ISO-8859-1	<ul style="list-style-type: none"> <li>• <i>headerCharset</i>: ISO-8859-1 &amp; Quoted-printable</li> <li>• <i>bodyCharset</i>: ISO-8859-1 y 8 bits</li> </ul>
mail mode UTF8	US-ASCII_UTF8_QP	<i>headerCharset</i> & <i>bodyCharset</i> : US-ASCII si es posible, de lo contrario UTF-8 & Quoted-printable (valor por defecto)
mail mode UTF8 in base64	US-ASCII_UTF8_B64	<i>headerCharset</i> & <i>bodyCharset</i> : US-ASCII si es posible, de lo contrario UTF-8 & base64

## .host

► Histórico

.host : Text

### Descripción

La propiedad `.host` contiene el nombre o la dirección IP del servidor local. Se utiliza para las transacciones de correo (SMTP, POP3, IMAP).

## .keepAlive

► Histórico

.keepAlive : Boolean

### Descripción

La propiedad `.keepAlive` contiene True si la conexión SMTP debe permanecer activa hasta que el objeto sea destruido `transporter` y False si no. Por defecto, si la propiedad `keepAlive` no se ha definido en el objeto `server` (que permite crear el objeto `transporter` vía el comando `SMTP New transporter`), es True.

La conexión SMTP se cierra automáticamente:

- cuando el objeto `transporter` se destruye si la `.keepAlive` es true,
- después de cada ejecución de la función `.send()` si la propiedad `.keepAlive` es false.

## .logFile

► Histórico

.logFile : Text

### Descripción

La propiedad `.logFile` contiene la ruta del archivo de registro extendido definido (si existe) para la conexión de correo. Puede ser relativo (a la carpeta actual Logs) o absoluto.

A diferencia de los archivos de registro clásicos (habilitados mediante el comando `SET DATABASE PARAMETER`), los archivos de registro extendidos almacenan el contenido MIME de todos los correos enviados y no tienen ningún límite de tamaño. Para más información sobre los archivos de registro extendidos, consulte:

- Conexiones SMTP - [4DSMTPLLog.txt](#)
- \*\*Conexiones POP3 \*\* - [4DSMTPLLog.txt](#)
- Conexiones IMAP - [4DIMAPLog.txt](#)

## .port

► Histórico

.port : Integer

### Descripción

La propiedad `.port` contiene el número de puerto utilizado para las transacciones de correo. Por defecto, si la propiedad `port` no se ha definido en el objeto `server` (utilizado para crear el objeto transportador con `SMTP New transporter`, `POP3 New transporter`, `IMAP New transporter`), el puerto utilizado es:

- SMTP - 587
- POP3 - 995
- IMAP - 993

## .send()

► Histórico

.send( *mail* : Object ) : Object

Parámetros	Tipo		Descripción
mail	Objeto	->	Email a enviar
Resultado	Objeto	<-	Estado SMTP

### Descripción

La función `.send()` envía el objeto `mail object` al servidor SMTP definido en el objeto `transporter` y devuelve un objeto estado.

El objeto `transporter` debe haberse creado ya con la función `SMTP New transporter`.

El método crea la conexión SMTP si no está ya activa. Si la propiedad `.keepAlive` del objeto `transporter` es false, la conexión SMTP se cierra automáticamente tras la ejecución del `.send()`. Para más información, consulte la descripción del comando `SMTP New transporter`.

En `mail`, pase un objeto `Email` válido a enviar. Las propiedades origen (de dónde viene el correo electrónico) y destino (uno o varios destinatarios) deben incluirse, el resto de propiedades son opcionales.

### Objeto devuelto

La función devuelve un objeto que describe el estado SMTP de la operación. Este objeto puede contener las siguientes propiedades:

Propiedad	Tipo	Descripción
success	booleano	True si el envío tiene éxito, false en caso contrario
status	number	Código de estado devuelto por el servidor SMTP (0 en caso de un problema no relacionado con el procesamiento del correo)
statusText	texto	Mensaje de estado devuelto por el servidor SMTP

En caso de un problema no relacionado con el procesamiento SMTP (por ejemplo, si falta una propiedad obligatoria en el correo), 4D genera un error que puede interceptar utilizando un método instalado por el comando `ON ERR CALL`. Utilice el comando `GET LAST ERROR STACK` para obtener información sobre el error.

En este caso, el objeto estado resultante contiene los siguientes valores:

Propiedad	Valor
success	False
status	0
statusText	"Failed to send email"

## .sendTimeOut

► Histórico

.sendTimeOut : Integer

### Descripción

La propiedad `.sendTimeOut` contiene el tiempo máximo de espera (en segundos) de una llamada a `.send()` antes de que se produzca un timeout. Por defecto, si la propiedad `.sendTimeOut` no se ha definido en el objeto `server`, se utiliza el valor 100.

## .user

► Histórico

.user : Text

### Descripción

La propiedad `.user` contiene el nombre de usuario utilizado para la autenticación en el servidor de correo.

# SystemWorker

System workers allow the 4D code to call any external process (a shell command, PHP, etc.) on the same machine. Los trabajadores del sistema se llaman de forma asíncrona. By using callbacks, 4D makes it possible to communicate both ways.

The `SystemWorker` class is available from the `4D` class store.

## Ejemplo

```
// Windows example to get access to the ipconfig information
var $myWinWorker : 4D.SystemWorker
var $ipConfig : Text
$myWinWorker:= 4D.SystemWorker.new("ipconfig")
$ipConfig:=$myWinWorker.wait(1).response //timeout 1 second

// macOS example to change the permissions for a file on macOS
// chmod is the macOS command used to modify file access
var $myMacWorker : 4D.SystemWorker
$myMacWorker:= 4D.SystemWorker.new("chmod +x /folder/myfile.sh")
```

## Resumen

## `4D.SystemWorker.new ( commandLine : Text { ; options : Object } ) : 4D.SystemWorker`

creates and returns a `4D.SystemWorker` object that will execute the `commandLine` you passed as parameter to launch an external process

### `.closeInput()`

closes the input stream (`stdin`) of the external process

### `.commandLine : Text`

contains the command line passed as parameter to the `new()` function

### `.currentDirectory : 4D.Folder`

contains the working directory in which the external process is executed

### `.dataType : Text`

contains the type of the response body content

### `.encoding : Text`

contains the encoding of the response body content

### `.errors : Collection`

contains a collection of 4D errors in case of execution error(s)

### `.exitCode : Integer`

contains the exit code returned by the external process

### `.hideWindow : Boolean`

can be used to hide the window of the DOS console or the window of the launched executable ( Windows only)

### `.pid : Integer`

contains the process unique identifier of the external process at the system level

### `.postMessage( message : Text)`

### `.postMessage( messageBLOB : Blob)`

allows you to write on the input stream (`stdin`) of the external process

### `.response : Text`

### `.response : Blob`

contains the concatenation of all data returned once the request is terminated

### `.responseError : Text`

contains the concatenation of all the errors returned, once the request is terminated

### `.terminate()`

forces the `SystemWorker` to terminate its execution

### `.terminated : Boolean`

contains true if the external process is terminated

### `.timeout : Integer`

contains the duration in seconds before the external process will be killed if it is still alive

### `.wait( {timeout : Real} ) : 4D.SystemWorker`

waits until the end of the `SystemWorker` execution or the specified `timeout`

## 4D.SystemWorker.new()

► Histórico

4D.SystemWorker.new ( *commandLine* : Text { ; *options* : Object } ) : 4D.SystemWorker

Parámetros	Tipo		Descripción
<i>commandLine</i>	Texto	->	Línea de comando a ejecutar
<i>options</i>	Objeto	->	Parámetros worker
<i>result</i>	4D.SystemWorker	<-	New asynchronous System worker or null if process not started

## Descripción

The `4D.SystemWorker.new()` function creates and returns a `4D.SystemWorker` object that will execute the *commandLine* you passed as parameter to launch an external process.

The returned system worker object can be used to post messages to the worker and get the worker output.

If an error occurs during the creation of the proxy object, the function returns a `null` object and an error is thrown.

In the *commandLine* parameter, pass the full path of the application's file to be executed (posix syntax), as well as any required arguments, if necessary. If you pass only the application name, 4D will use the `PATH` environment variable to locate the executable.

Warning: This function can only launch executable applications; it cannot execute instructions that are part of the shell (command interpreter). For example, under Windows it is not possible to use this command to execute the `dir` instruction.

## Objeto *options*

En el parámetro *options*, pase un objeto que pueda contener las siguientes propiedades:

Propiedad	Tipo	Por defecto	Descripción
onResponse	Formula	indefinido	Retrollamada para los mensajes del system worker. This callback is called once the complete response is received. Recibe dos objetos como parámetros (ver abajo)
onData	Formula	indefinido	Retrollamada para los datos del system worker. This callback is called each time the system worker receives data. Recibe dos objetos como parámetros (ver abajo)
onDataError	Formula	indefinido	Callback for the external process errors ( <code>stderr</code> of the external process). Recibe dos objetos como parámetros (ver abajo)
onError	Formula	indefinido	Callback for execution errors, returned by the system worker in case of unusual runtime conditions (system errors). Recibe dos objetos como parámetros (ver abajo)
onTerminate	Formula	indefinido	Retrollamada cuando el proceso externo se termina. Recibe dos objetos como parámetros (ver abajo)
timeout	Número	indefinido	Time in seconds before the process is killed if it is still alive
dataType	Texto	"text"	Type of the response body content. Valores posibles: "text" (por defecto), "blob".
encoding	Texto	"UTF-8"	Sólo si <code>dataType="text"</code> . Codificación del contenido del cuerpo de la respuesta. For the list of available values, see the <a href="#">CONVERT FROM TEXT</a> command description
variables	Objeto		Sets custom environment variables for the system worker. Syntax: <code>variables.key=value</code> , where <code>key</code> is the variable name and <code>value</code> its value. Values are converted into strings when possible. El valor no puede contener un '='. If not defined, the system worker inherits from the 4D environment.
currentDirectory	Folder		Directorio de trabajo en el que se ejecuta el proceso
hideWindow	Booleano	true	(Windows) Hide the application window (if possible) or the Windows console

All callback functions receive two object parameters. Su contenido depende de la retrollamada:

Parámetros	Tipo	onResponse	onData	onDataError	onError	onTerminate
\$param1	Objeto	SystemWorker	SystemWorker	SystemWorker	SystemWorker	SystemWorker
\$param2.type	Texto	"response"	"data"	"error"	"error"	"termination"
\$param2.data	Text o Blob		datos recibidos	error data		

Esta es la secuencia de llamadas de retorno:

1. `onData` and `onDataError` are executed one or several times
2. if called, `onError` is executed once (stops the system worker processing)
3. if no error occurred, `onResponse` is executed once
4. `onTerminate` se ejecuta siempre

Valor devuelto

The function returns a system worker object on which you can call functions and properties of the SystemWorker class.

## Ejemplos en Windows

1. Para abrir el Bloc de notas y abrir un documento específico:

```

var $sw : 4D.SystemWorker
var $options : Object
$options:=New object
$options.hideWindow:= False

$sw:=4D.SystemWorker.new ("C:\\\\WINDOWS\\\\notepad.exe C:\\\\Docs\\\\new folder\\\\res.txt";$options)

```

2. Ejecutar npm install en la consola:

```

var $folder : 4D.Folder
var $options : Object
var $worker : 4D.SystemWorker

$folder:=Folder(fk database folder)
$options:=New object
$options.currentDirectory:=$folder
$options.hideWindow:=False

$worker:=4D.SystemWorker.new("cmd /c npm install";$options)

```

3. To launch the Microsoft® Word® application and open a specific document:

```

$mydoc:="C:\\Program Files\\Microsoft Office\\Office15\\WINWORD.EXE C:\\Tempo\\output.txt"
var $sw : 4D.SystemWorker
$sw:=4D.SystemWorker.new($mydoc)

```

4. To launch a command with the current directory and post a message:

```

var $param : Object
var $sys : 4D.SystemWorker

$param:=New object
$param.currentDirectory:=Folder(fk database folder)
$sys:=4D.SystemWorker.new("git commit -F -";$param)
$sys.postMessage("This is a postMessage")
$sys.closeInput()

```

5. To allow the user to open an external document on Windows:

```

$docname:=Select document("");".*";"Elija el archivo a abrir";0
If(OK=1)
    var $sw : 4D.SystemWorker
    $sw:=4D.SystemWorker.new("cmd.exe /C start \"\" \"$docname\"\"")
End if

```

### Ejemplos en macOS

1. Edit a text file ( `cat` is the macOS command used to edit files). In this example, the full access path of the command is passed:

```

var $sw : 4D.SystemWorker
$sw:=4D.SystemWorker.new("/bin/cat /folder/myfile.txt")
$sw.wait() //synchronous execution

```

2. To launch an independent "graphic" application, it is preferable to use the `open` system command (in this case, the code has the same effect as double-clicking the application):

```

var $sw : 4D.SystemWorker
$sw:=4D.SystemWorker.new ("open /Applications/Calculator.app")

```

3. To get the contents of the "Users" folder (ls -l is the macOS equivalent of the dir command in DOS).

```

var $systemworker : 4D.SystemWorker
var $output : Text
var $errors : Collection

$systemworker:=4D.SystemWorker.new("/bin/ls -l /Users ")
$systemworker.wait(5)
$output:=$systemworker.response
$errors:=$systemworker.errors

```

4. Same command as above, but using a sample "Params" user class to show how to handle callback functions:

```

var $systemworker : 4D.SystemWorker
$systemworker:=4D.SystemWorker.new("/bin/ls -l /Users ";cs.Params.new())

// "Params" class

Class constructor
    This.dataType:="text"
    This.data:=""
    This.dataError:=""

Function onResponse($systemWorker : Object)
    This._createFile("onResponse"; $systemWorker.response)

Function onData($systemWorker : Object; $info : Object)
    This.data+=$info.data
    This._createFile("onData";this.data)

Function onDataError($systemWorker : Object; $info : Object)
    This.dataError+=$info.data
    This._createFile("onDataError";this.dataError)

Function onTerminate($systemWorker : Object)
    var $textBody : Text
    $textBody:="Response: "+$systemWorker.response
    $textBody+="ResponseError: "+$systemWorker.responseError
    This._createFile("onTerminate"; $textBody)

Function _createFile($title : Text; $textBody : Text)
    TEXT TO DOCUMENT(Get 4D folder(Current resources folder)+$title+".txt"; $textBody)

```

## .closeInput()

► Histórico

.closeInput() | Parámetros | Tipo | Descripción | | ----- | ---- |::| ----- | | | | No requiere ningún parámetro |

### Descripción

The `.closeInput()` function closes the input stream (`stdin`) of the external process.

When the executable waits for all data to be received through `postMessage()`, `.closeInput()` is useful to indicate to the executable that data sending is finished and that it can proceed.

### Ejemplo

```
// Create some data to gzip
var $input;$output : Blob
var $gzip : Text
TEXT TO BLOB("Hello, World!";$input)
$gzip:="\"C:\\Program Files (x86)\\GnuWin32\\bin\\gzip.exe\""

// Create an asynchronous system worker
var $worker : 4D.SystemWorker
$worker:= 4D.SystemWorker.new($gzip;New object("dataType";"blob"))

// Send the compressed file on stdin.
$worker.postMessage($input)
// Note that we call closeInput() to indicate we're done.
// gzip (and most program waiting data from stdin) will wait for more data until the input is explicitly closed
$worker.closeInput()
$worker.wait()

$output:=$worker.response
```

## .commandLine

.commandLine : Text

### Descripción

The `.commandLine` property contains the command line passed as parameter to the `new()` function.

Esta propiedad es de sólo lectura.

## .currentDirectory

.currentDirectory : 4D.Folder

### Descripción

The `.currentDirectory` property contains the working directory in which the external process is executed.

## .dataType

.dataType : Text

### Descripción

The `.dataType` property contains the type of the response body content. Valores posibles: "text" o "blob".

Esta propiedad es de sólo lectura.

## .encoding

`.encoding` : Text

Descripción

The `.encoding` property contains the encoding of the response body content. This property is only available if the `dataType` is "text".

Esta propiedad es de sólo lectura.

## .errors

`.errors` : Collection

Descripción

The `.errors` property contains a collection of 4D errors in case of execution error(s).

Each element of the collection is an object with the following properties:

Propiedad	Tipo	Descripción
<code>[]errorCode</code>	number	Código de error 4D
<code>[]message</code>	texto	Descripción del error 4D
<code>[ ].componentSignature</code>	texto	Firma del componente interno que ha devuelto el error

If no error occurred, `.errors` contains an empty collection.

## .exitCode

`.exitCode` : Integer

Descripción

The `.exitCode` property contains the exit code returned by the external process. If the process did not terminate normally, `exitCode` is *undefined*.

Esta propiedad es de sólo lectura.

## .hideWindow

`.hideWindow` : Boolean

Descripción

The `.hideWindow` property can be used to hide the window of the DOS console or the window of the launched executable (Windows only).

Esta propiedad es lectura-escritura.

## .pid

`.pid` : Integer

## Descripción

The `.pid` property contains the process unique identifier of the external process at the system level.

Esta propiedad es de sólo lectura.

## .postMessage()

`.postMessage( message : Text)`  
`.postMessage( messageBLOB : Blob)`

Parámetros	Tipo		Descripción
message	Texto	->	Text to write on the input stream ( <code>stdin</code> ) of the external process
messageBLOB	Blob	->	Bytes escritos en el flujo de entrada

## Descripción

The `.postMessage()` function allows you to write on the input stream (`stdin`) of the external process. In the `message` parameter, pass the text to write in `stdin`.

The `.postMessage()` function also accepts a Blob type value in `messageBLOB` to pass in `stdin`, so that you can post binary data.

You can use the `.dataType` property of the [options object](#) to make response body return Blob values.

## .response

`.response : Text`  
`.response : Blob`

## Descripción

The `.response` property contains the concatenation of all data returned once the request is terminated, i.e. the full message received from the process output.

The type of the message is defined according to the `dataType` attribute.

Esta propiedad es de sólo lectura.

## .responseError

`.responseError : Text`

## Descripción

The `.responseError` property contains the concatenation of all the errors returned, once the request is terminated.

## .terminate()

`.terminate()` | Parámetros | Tipo | | Descripción | | ----- | ---- | ::| ----- | | | | | No requiere ningún parámetro |

## Descripción

The `.terminate()` function forces the `SystemWorker` to terminate its execution.

This function sends the instruction to terminate and give control back to the executing script.

## .terminated

.terminated : Boolean

### Descripción

The `.terminated` property contains true if the external process is terminated.

Esta propiedad es de sólo lectura.

## .timeout

.timeout : Integer

### Descripción

The `.timeout` property contains the duration in seconds before the external process will be killed if it is still alive.

Esta propiedad es de sólo lectura.

## .wait()

► Histórico

`.wait( {timeout : Real} ) : 4D.SystemWorker`

Parámetros	Tipo		Descripción
timeout	Real	->	Tiempo de espera (en segundos)
Resultado	4D.SystemWorker	<-	Objeto SystemWorker

### Descripción

The `.wait()` function waits until the end of the `SystemWorker` execution or the specified *timeout*.

In *timeout*, pass a value in seconds. The `SystemWorker` script will wait for the external process for the amount of time defined in the *timeout* parameter. If you omit the *timeout* parameter, the script execution will wait indefinitely.

Actually, `.wait()` waits until the end of processing of the `onTerminate` formula, except if the *timeout* is reached. If *timeout* is reached, the `SystemWorker` is not killed.

During a `.wait()` execution, callback functions are executed, especially callbacks from other events or from other `SystemWorker` instances. You can exit from a `.wait()` by calling `terminate()` from a callback.

Esta función devuelve el objeto SystemWorker.

Esta función no es necesaria si ha creado el `SystemWorker` de un proceso worker 4D.

# WebServer

La API de la clase `WebServer` permite iniciar y supervisar un servidor web para la aplicación principal (host), así como para cada componente alojado (ver la descripción general de [Objeto servidor web](#)). Esta clase está disponible en el almacén de clases de `4D`.

## Objeto servidor web

Los objetos del servidor web se instancian con el comando `WEB Server`.

Ofrecen las siguientes propiedades y funciones:

## Resumen

<code>.accessKeyDefined : Boolean</code>	true si se define una llave de acceso en la configuración del servidor web
<code>.certificateFolder : Text</code>	donde se encuentran los archivos de los certificados
<code>.characterSet : Number</code> <code>.characterSet : Text</code>	conjunto de caracteres que el Servidor Web <code>4D</code> debe utilizar para comunicarse con los navegadores que se conectan a la aplicación
<code>.cipherSuite : Text</code>	utilizada para el protocolo seguro
<code>.CORSEnabled : Boolean</code>	CORS ( <i>Cross-origin resource sharing</i> ) para el servidor web
<code>.CORSSettings : Collection</code>	lista de hosts y métodos permitidos para el servicio CORS
<code>.debugLog : Number</code>	estado del archivo de registro de peticiones HTTP
<code>.defaultHomepage : Text</code>	nombre de la página de inicio por defecto
<code>.HSTSEnabled : Boolean</code>	HTTP Strict Transport Security (HSTS)
<code>.HSTSMaxAge : Number</code>	máximo de tiempo (en segundos) que HSTS está activo para cada nueva conexión cliente
<code>.HTTPCompressionLevel : Number</code>	nivel de compresión para todos los intercambios HTTP comprimidos para el servidor HTTP <code>4D</code> (peticiones del cliente o respuestas del servidor)
<code>.HTTPCompressionThreshold : Number</code>	umbral de tamaño (bytes) para las solicitudes por debajo del cual los intercambios no deben ser comprimidos
<code>.HTTPEnabled : Boolean</code>	HTTP

.HTTPPort : Number número de puerto IP de escucha para HTTP
.HTTPTrace : Boolean activación de <code>HTTP TRACE</code>
.HTTPSEnabled : Boolean
.HTTPSPort : Number número de puerto IP de escucha para HTTPS
.inactiveProcessTimeout : Number duración (en minutos) de los procesos de sesión heredados inactivos
.inactiveSessionTimeout : Number duración (en minutos) de las sesiones heredadas inactivas (duración establecida en la cookie)
.IPAddressToListen : Text dirección IP en la que el servidor web 4D recibirá las peticiones HTTP
.isRunning : Boolean estado de ejecución del servidor web
.keepSession : Boolean True si las sesiones heredadas están activadas en el servidor web, False en caso contrario
.logRecording : Number log (logweb.txt)
.maxConcurrentProcesses : Number número máximo de procesos web concurrentes que soporta el servidor web
.maxRequestSize : Number tamaño máximo (en bytes) de las peticiones HTTP entrantes (POST) que el servidor web puede procesar
.maxSessions : Number número máximo de sesiones simultáneas legacy
.minTLSVersion : Number versión TLS mínima aceptada para las conexiones
.name : Text nombre de la aplicación del servidor web
.openSSLVersion : Text versión de la librería OpenSSL utilizada
.perfectForwardSecrecy : Boolean PFS en el servidor
.rootFolder : Text ruta de la carpeta raíz del servidor web
.scalableSession : Boolean True si las sesiones escalables son utilizadas en el servidor web, False en caso contrario

```

.sessionCookieDomain : Text
  "domain" de la cookie de sesión| | .sessionCookieName : Text
  nombre de la cookie utilizada para almacenar el ID de sesión| | .sessionCookiePath : Text
  "path" de la cookie de sesión| | .sessionCookieSameSite : Text
  "SameSite"| | .sessionIPAddressValidation : Boolean
    validación de la dirección IP para las cookies de sesión| | .start() : Object
.start( settings : Object ) : Object
  inicia el servidor web sobre el que se aplica| | .stop()
  detiene el servidor web sobre el que se aplica

```

## WEB Server

► Histórico

WEB Server : 4D.WebServer

WEB Server( option : Integer ) : 4D.WebServer

Parámetros	Tipo		Descripción
option	Integer	->	Servidor web a obtener (por defecto si se omite = Web server database )
Resultado	4D.WebServer	<-	Objeto servidor web

La función `.start()` inicia el servidor web sobre el que se aplica, utilizando las propiedades establecidas en el parámetro opcional del objeto *settings*.

Por defecto, si se omite el parámetro *option*, el comando devuelve una referencia al servidor web de la base de datos, es decir, al servidor web por defecto. Para designar el servidor web a devolver, puede pasar una de las siguientes constantes en el parámetro *option*:

Constante	Valor	Comentario
Web server database	1	Servidor web de la base actual (por defecto si se omite)
Web server host database	2	Servidor web de la base local de un componente
Web server receiving request	3	Servidor web que ha recibido la solicitud (servidor web objetivo)

El objeto servidor web devuelto contiene los valores actuales de las propiedades del servidor web.

### Ejemplo

El objeto servidor web devuelto contiene los valores actuales de las propiedades del servidor web.

```

// Método de un componente
var $hostWS : 4D.WebServer
$hostWS:=WEB Server(Web server host database)
If($hostWS.isRunning)
  ...
End if

```

## WEB Server list

► Histórico

WEB Server list : Collection

Parámetros	Tipo		Descripción
Resultado	Collection	<-	Colección de los objetos del servidor web disponibles

El nombre de la aplicación del servidor web.

Queremos saber cuántos servidores web en funcionamiento hay disponibles:

- un servidor web para la base de datos del host (servidor web por defecto)
- un servidor web para cada componente.

Una aplicación 4D puede contener de uno a varios servidores web:

El objeto servidor web por defecto es cargado automáticamente por 4D al inicio. Por otro lado, cada componente servidor web que se quiera utilizar debe ser instanciado utilizando el comando `WEB Server`.

El comando `WEB Server list` devuelve todos los servidores web disponibles, estén o no en funcionamiento.

### Ejemplo

Puede utilizar la propiedad `.name` del objeto servidor web para identificar el proyecto o componente al que está unido cada objeto servidor web de la lista.

```
var $wSList : Collection
var $vRun : Integer

$wSList:=WEB Server list
$vRun:=$wSList.countValues(True;"isRunning")
ALERT(String($vRun)+" web server(s) running on "+String($wSList.length)+" available.")
```

## .accessKeyDefined

`.accessKeyDefined` : Boolean

La propiedad `.accessKeyDefined` contiene true si se define una llave de acceso en la configuración del servidor web. Esta propiedad es utilizada por el servidor web de WebAdmin para validar la configuración de seguridad de la interfaz de administración.

## .certificateFolder

`.certificateFolder` : Text

Ruta de la carpeta donde se encuentran los archivos de los certificados. La ruta se formatea en la ruta completa POSIX utilizando filesystems. Cuando se utiliza esta propiedad en el parámetro `settings` de la función `.start()`, puede ser un objeto `Folder`.

## .characterSet

`.characterSet` : Number  
`.characterSet` : Text

El conjunto de caracteres que el Servidor Web 4D debe utilizar para comunicarse con los navegadores que se conectan a la aplicación. El valor por defecto depende del lenguaje del sistema operativo. Puede ser un entero MIBEnum o una cadena Name, identificadores definidos por IANA. Aquí está la lista de identificadores correspondientes a los conjuntos de caracteres soportados por el servidor web 4D:

- 4 = ISO-8859-1
- 12 = ISO-8859-9

- 13 = ISO-8859-10
- 17 = Shift-JIS
- 2024 = Windows-31J
- 2026 = Big5
- 38 = euc-kr
- 106 = UTF-8
- 2250 = Windows-1250
- 2251 = Windows-1251
- 2253 = Windows-1253
- 2255 = Windows-1255
- 2256 = Windows-1256

## .cipherSuite

.cipherSuite : Text

La lista de cifrado utilizada para el protocolo seguro. Define la prioridad de los algoritmos de cifrado implementados por el servidor web de 4D. Puede ser una secuencia de cadenas separadas por dos puntos (por ejemplo "ECDHE-RSA-AES128-..."). Ver la [página de cifrados](#) en el sitio OpenSSL.

## .CORSEnabled

.CORSEnabled : Boolean

El estado del servicio CORS (*Cross-origin resource sharing*) para el servidor web. Por razones de seguridad, las peticiones "cross-domain" están prohibidas por defecto a nivel del navegador. Cuando está habilitado (True), las llamadas XHR (por ejemplo, peticiones REST) de páginas web fuera del dominio pueden ser permitidas en su aplicación (necesita definir la lista de direcciones permitidas en la lista de dominios CORS, ver `CORSSettings` abajo). Cuando se desactiva (False, por defecto), se ignoran todas las peticiones cruzadas enviadas con CORS. Cuando se activa (True) y un dominio o método no permitido envía una solicitud de sitio cruzado, se rechaza con una respuesta de error "403 - prohibido".

Por defecto: False (desactivado)

Para más información sobre CORS, consulte la página [Cross-origin resource sharing page](#) en Wikipedia.

## .CORSSettings

.CORSSettings : Collection

Una lista de hosts y métodos permitidos para el servicio CORS (ver la propiedad `CORSEnabled`). Cada objeto debe contener una propiedad host y, opcionalmente, una propiedad methods:

- host (texto, obligatorio): nombre de dominio o dirección IP desde donde las páginas externas pueden enviar solicitudes de datos al Servidor a través de CORS. Se pueden añadir múltiples atributos de dominio para crear una lista blanca. Si `host` no está presente o está vacío, el objeto se ignora. Se soportan varias sintaxis:
  - 192.168.5.17:8081
  - 192.168.5.17
  - 192.168.\*
  - 192.168.\*:8081
  - <http://192.168.5.17:8081>
  - [http://\\*.myDomain.com](http://*.myDomain.com)
  - <http://myProject.myDomain.com>
  - \*.myDomain.com
  - myProject.myDomain.com
  - \*
- methods (texto, opcional): método(s) HTTP aceptado(s) para el host CORS correspondiente. Separe cada método

con un ";" (por ejemplo: "post;get"). Si *methods* está vacío, null o indefinido, todos los métodos están activos.

## .debugLog

.debugLog : Number

El estado del archivo de registro de peticiones HTTP (HTTPDebugLog\_nn.txt, almacenado en la carpeta "Logs" de la aplicación -- nn es el número de archivo).

- 0 = desactivado
- 1 = activado sin partes del cuerpo (en este caso se suministra el tamaño del cuerpo)
- 3 = activado con las partes del cuerpo en respuesta únicamente
- 5 = activado con las partes del cuerpo en petición únicamente
- 7 = activado con las partes del cuerpo en respuesta y petición

## .defaultHomepage

.defaultHomepage : Text

El nombre de la página de inicio por defecto o "" para no enviar la página de inicio personalizada.

## .HSTSEnabled

.HSTSEnabled : Boolean

El estado HTTP Strict Transport Security (HSTS). HSTS permite al servidor web declarar que los navegadores sólo deben interactuar con él a través de conexiones HTTPS seguras. Los navegadores registrarán la información HSTS la primera vez que reciban una respuesta del servidor web, luego cualquier solicitud HTTP futura se transformará automáticamente en solicitudes HTTPS. El tiempo que esta información es almacenada por el navegador se especifica con la propiedad `HSTSMaxAge`. HSTS requiere que HTTPS esté activado en el servidor. HTTP también debe estar activado para permitir las conexiones cliente iniciales.

## .HSTSMaxAge

.HSTSMaxAge : Number

El máximo de tiempo (en segundos) que HSTS está activo para cada nueva conexión cliente. Esta información se almacena del lado del cliente durante el tiempo especificado.

Valor por defecto: 63072000 (2 años).

## .HTTPCompressionLevel

.HTTPCompressionLevel : Number

El nivel de compresión para todos los intercambios HTTP comprimidos para el servidor HTTP 4D (peticiones del cliente o respuestas del servidor). Este selector permite optimizar los intercambios priorizando la velocidad de ejecución (menos compresión) o la cantidad de compresión (menos velocidad).

Valores posibles:

- 1 a 9 (donde 1 es la compresión más rápida y 9 la más alta).
- -1 = definir un compromiso entre la velocidad y la tasa de compresión.

Valores posibles:

## .HTTPCompressionThreshold

.HTTPCompressionThreshold : Number

El umbral de tamaño (bytes) para las solicitudes por debajo del cual los intercambios no deben ser comprimidos. Este parámetro es útil para evitar la pérdida de tiempo de la máquina al comprimir los intercambios pequeños.

Umbral de compresión por defecto = 1024 bytes

## .HTTPEnabled

.HTTPEnabled : Boolean

El estado del protocolo HTTP.

## .HTTPPort

.HTTPPort : Number

El estado del protocolo HTTPS.

El número de puerto IP de escucha para HTTP.

## .HTTPTrace

.HTTPTrace : Boolean

La activación de `HTTP TRACE`. Por razones de seguridad, por defecto el servidor web rechaza las peticiones `HTTP TRACE` con un error 405. Cuando se activa, el servidor web responde a las peticiones `HTTP TRACE` con la línea de petición, el encabezado y el cuerpo.

## .HTTPSEnabled

.HTTPSEnabled : Boolean El estado de ejecución del servidor web.

## .HTTPSPort

.HTTPSPort : Number La disponibilidad de PFS en el servidor.

El número de puerto IP de escucha para HTTPS.

## .inactiveProcessTimeout

.inactiveProcessTimeout : Number

Esta propiedad no se devuelve en [modo sesiones escalables](#).

La duración (en minutos) de los procesos de sesión heredados inactivos. Al final del tiempo de espera, el proceso se mata en el servidor, se llama al método base `On Web Legacy Close Session` y se destruye el contexto de la sesión heredada.

Por defecto = 480 minutos

## .inactiveSessionTimeout

.inactiveSessionTimeout : Number

Esta propiedad no se devuelve en [modo sesiones escalables](#).

La duración (en minutos) de las sesiones heredadas inactivas (duración establecida en la cookie). Al final de este

periodo, la cookie de sesión expira y deja de ser enviada por el cliente HTTP.

Por defecto = 480 minutos

## .IPAddressToListen

.IPAddressToListen : Text

La dirección IP en la que el servidor web 4D recibirá las peticiones HTTP. Por defecto, no se define ninguna dirección específica. Se soportan tanto los formatos de cadena IPv6 como los IPv4.

## .isRunning

.isRunning : Boolean

*Propiedad de sólo lectura*

La función `.stop()` detiene el servidor web sobre el que se aplica.

## .keepSession

.keepSession : Boolean

True si las sesiones heredadas están activadas en el servidor web, False en caso contrario.

Ver también:

[.scalableSession](#)

## .logRecording

.logRecording : Number

El valor de registro log (logweb.txt).

- 0 = No registrar (por defecto)
- 1 = Registro en formato CLF
- 2 = Registro en formato DLF
- 3 = Registro en formato ELF
- 4 = Registro en formato WLF

## .maxConcurrentProcesses

.maxConcurrentProcesses : Number

El número máximo de procesos web concurrentes que soporta el servidor web. Cuando se alcance este número (menos uno), 4D no creará ningún otro proceso y devolverá el estado HTTP 503 - Servicio no disponible a todas las nuevas peticiones.

Valores posibles: 500000 - 2147483647

Valores posibles: 500000 - 2147483648

## .maxRequestSize

.maxRequestSize : Number

El tamaño máximo (en bytes) de las peticiones HTTP entrantes (POST) que el servidor web puede procesar. Pasar el valor máximo (2147483647) significa que, en la práctica, no se define ningún límite. Este límite se utiliza para evitar la saturación del servidor web debido a peticiones entrantes demasiado grandes. Si una petición alcanza este límite, el

servidor web la rechaza.

Valores posibles: 500000 - 2147483647

## .maxSessions

.maxSessions : Number

Esta propiedad no se devuelve en [modo sesiones escalables](#).

El número máximo de sesiones simultáneas legacy. Cuando se alcanza el límite, se cierra la sesión heredada más antigua (y se llama al método base `On Web Legacy Close Session`) si el servidor web necesita crear una nueva. El número de sesiones heredadas simultáneas no puede superar el número total de procesos web (propiedad `maxConcurrentProcesses`, 100 por defecto)

## .minTLSVersion

.minTLSVersion : Number

La versión TLS mínima aceptada para las conexiones. Se rechazarán los intentos de conexión de clientes que sólo soporten versiones inferiores a la mínima.

Valores posibles:

- 1 = TLSv1\_0
- 2 = TLSv1\_1
- 3 = TLSv1\_2 (por defecto)
- 4 = TLSv1\_3

Valores posibles:

## .name

.name : Text

*Propiedad de sólo lectura*

El nombre de la cookie utilizada para almacenar el ID de sesión.

## .openSSLVersion

.openSSLVersion : Text

*Propiedad de sólo lectura*

La versión de la librería OpenSSL utilizada.

## .perfectForwardSecrecy

.perfectForwardSecrecy : Boolean

*Propiedad de sólo lectura*

La disponibilidad de PFS en el servidor.

## .rootFolder

.rootFolder : Text

La ruta de la carpeta raíz del servidor web. La ruta se formatea en la ruta completa POSIX utilizando filesystems. Cuando se utiliza esta propiedad en el parámetro `settings`, puede ser un objeto `Folder`.

## .scalableSession

`.scalableSession` : Boolean

True si las sesiones escalables son utilizadas en el servidor web, False en caso contrario.

Ver también:

[.keepSession](#)

## .sessionCookieDomain

`.sessionCookieDomain` : Text

El campo "domain" de la cookie de sesión. Se utiliza para controlar el alcance de las cookies de sesión. Si define, por ejemplo, el valor "/\*.4d.fr" para este selector, el cliente sólo enviará una cookie cuando la solicitud se dirija al dominio ".4d.fr", lo que excluye a los servidores que alojan datos estáticos externos.

## .sessionCookieName

`.sessionCookieName` : Text

Ver la descripción de [Session Cookie SameSite](#) para obtener información detallada.

*Propiedad de sólo lectura*

## .sessionCookiePath

`.sessionCookiePath` : Text

El campo "path" de la cookie de sesión. Se utiliza para controlar el alcance de las cookies de sesión. Si define, por ejemplo, el valor "/4DACTION" para este selector, el cliente sólo enviará una cookie para las peticiones dinámicas que empiecen por 4DACTION, y no para las imágenes, páginas estáticas, etc.

## .sessionCookieSameSite

► Histórico

`.sessionCookieSameSite` : Text

El valor de la cookie de sesión "SameSite". Valores posibles (utilizando constantes):

Constante	Valor	Descripción
Web SameSite Strict	"Strict"	Valor por defecto - Las cookies sólo se envían en un contexto de primera parte
Web SameSite Lax	"Lax"	Las cookies también se envían en las sub-solicitudes entre sitios, pero sólo cuando un usuario está navegando hacia el sitio de origen (es decir, cuando sigue un enlace).
Web SameSite None	"None"	Las cookies se envían en todos los contextos, es decir, en las respuestas a las solicitudes de primera parte y de origen cruzado.

Ver la descripción de [Session Cookie SameSite](#) para obtener información detallada.

## .sessionIPAddressValidation

.sessionIPAddressValidation : Boolean

Esta propiedad no se utiliza en el modo [sesiones escalables](#) (no hay validación de la dirección IP).

La validación de la dirección IP para las cookies de sesión. Por razones de seguridad, por defecto el servidor web comprueba la dirección IP de cada solicitud que contiene una cookie de sesión y la rechaza si esta dirección no coincide con la dirección IP utilizada para crear la cookie. En algunas aplicaciones específicas, es posible que desee desactivar esta validación y aceptar las cookies de sesión, incluso cuando sus direcciones IP no coinciden. Por ejemplo, cuando los dispositivos móviles cambian entre las redes WiFi y 3G/4G, su dirección IP cambiará. En este caso, puede permitir que los clientes puedan seguir utilizando sus sesiones web incluso cuando las direcciones IP cambien (esta configuración reduce el nivel de seguridad de su aplicación).

## .start()

► Histórico

.start() : Object

.start( *settings* : Object ) : Object

Parámetros	Tipo		Descripción
parámetros	Objeto	->	Parámetros del servidor web a definir al inicio
Resultado	Objeto	<-	Estado del inicio del servidor web

El comando `WEB Server` devuelve el objeto servidor web por defecto, o el objeto servidor web definido a través del parámetro *option*.

El servidor web se inicia con la configuración por defecto definida en el archivo de configuración del proyecto o (sólo en la base host) utilizando el comando `WEB SET OPTION`. Sin embargo, utilizando el parámetro *settings*, se pueden definir propiedades personalizadas para la sesión del servidor web.

Todas las configuraciones de los [objetos del Servidor Web](#) pueden personalizarse, excepto las propiedades de sólo lectura (`.isRunning`, `.name`, `.openSSLVersion`, `.perfectForwardSecrecy` y `.sessionCookieName`).

Todas las configuraciones de los [objetos del Servidor Web](#) pueden personalizarse, excepto las propiedades de sólo lectura (`.isRunning`, `.name`, `.openSSLVersion`, `.perfectForwardSecrecy` y `.sessionCookieName`).

### Objeto devuelto

La función devuelve un objeto que describe el estado de lanzamiento del servidor web. Este objeto puede contener las siguientes propiedades:

Propiedad		Tipo	Descripción
success		Booleano	True si el servidor web se ha iniciado correctamente, False en caso contrario
errors		Collection	Pila de errores 4D (no se devuelve si el servidor web se inició con éxito)
	[].errCode	Número	Código de error 4D
	[].message	Texto	Descripción del error 4D
	[].componentSignature	Texto	Firma del componente interno que ha devuelto el error

Si el servidor web ya fue lanzado, se devuelve un error.

### Ejemplo

```
var $settings;$result : Object
var $webServer : 4D.WebServer

$settings:=New object("HTTPPort";8080;"defaultHomepage";"myAdminHomepage.html")

$webServer:=WEB Server
$result:=$webServer.start($settings)
If($result.success)
//...
End if
```

## .stop()

► Histórico

.stop()

Parámetros	Tipo	Descripción	-----	----	-----			No requiere ningún parámetro
------------	------	-------------	-------	------	-------	--	--	------------------------------

La función `.stop()` detiene el servidor web sobre el que se aplica.

Si el servidor web se ha iniciado, todas las conexiones y procesos web se cierran, una vez que las peticiones actualmente gestionadas han finalizado. Si el servidor web no se ha iniciado, el método no hace nada.

Esta función reinicia los parámetros web personalizados definidos para la sesión mediante el parámetro `settings` de la función `.start()`, si la hubiera.

### Ejemplo

Para detener el servidor web de la base de datos:

```
var $webServer : 4D.WebServer

$webServer:=WEB Server(Web server database)
$webServer.stop()
```

# ZIPArchive

Un archivo ZIP 4D es un objeto `File` o `Folder` que contiene uno o más archivos o carpetas, que se comprimen para ser más pequeños que su tamaño original. Estos archivos se crean con una extensión ".zip" y pueden utilizarse para ahorrar espacio en el disco o transferir archivos a través de medios que pueden tener limitaciones de tamaño (por ejemplo, el correo electrónico o la red).

- Cree un archivo ZIP 4D con el comando [ZIP Create archive](#).
- Las instancias `ZIPFile` y `ZIPFolder` de 4D están disponibles vía la propiedad `root` (`ZIPFolder`) del objeto devuelto por el comando [ZIP Read archive](#).

## Ejemplo

Para recuperar y ver el contenido de un objeto ZIP file:

```
var $path; $archive : 4D.File
var $zipFile : 4D.ZipFile
var $zipFolder : 4D.ZipFolder
var $txt : Text

$path:=Folder(fk desktop folder).file("MyDocs/Archive.zip")
$archive:=ZIP Read archive($path)
$zipFolder:=$archive.root // guardar la carpeta principal del zip
$zipFile:=$zipFolder.files()[0] //leer la primera carpeta comprimida

If($zipFile.extension=".txt")
    $txt:=$zipFile.getText()
End if
```

## Resumen

<code>.root : 4D.ZipFolder</code> una carpeta virtual que permite acceder al contenido del archivo ZIP
---

## ZIP Create archive

► Histórico

`ZIP Create archive ( fileToZip : 4D.File ; destinationFile : 4D.File ) : Object`

`ZIP Create archive ( folderToZip : 4D.Folder ; destinationFile : 4D.File { ; options : Integer } ) : Object`

`ZIP Create archive ( zipStructure : Object ; destinationFile : 4D.File ) : Object`

Parámetros	Tipo		Descripción
fileToZip	4D.File	->	Objeto archivo o carpeta a comprimir
folderToZip	4D.Folder	->	Objeto archivo o carpeta a comprimir
zipStructure	Objeto	->	Objeto archivo o carpeta a comprimir
destinationFile	4D.File	->	Archivo de destino del archivo
options	Integer	->	Opción <code>folderToZip</code> : <code>ZIP Without enclosing folder</code>
Resultado	Objeto	<-	Objeto de estado

Descripción

El comando `ZIP Create archive` crea un objeto archivo ZIP comprimido y devuelve el estado de la operación.

Puede pasar un objeto `4D.File`, `4D.Folder`, o una estructura zip como primer parámetro:

- `fileToZip`: simplemente pase un `4D.File` a comprimir.
- `folderToZip`: pase un `4D.Folder` a comprimir. En este caso, el parámetro `options` permite comprimir sólo el contenido de la carpeta (es decir, excluir la carpeta padre). Por defecto, `ZIP Create archive` comprimirá la carpeta y su contenido, de modo que la operación de descompresión volverá a crear una carpeta. Si desea que la operación de descompresión restaure sólo el contenido de la carpeta, pase la constante `ZIP Without enclosing folder` en el parámetro `options`.
- `zipStructure`: pase un objeto que describa el objeto ZIP archivo. Las siguientes propiedades están disponibles para definir la estructura:

Propiedad	Tipo	Descripción												
compression	Integer	<ul style="list-style-type: none"><li>• <code>ZIP Compression standard</code> : Reducir la compresión (por defecto)</li><li>• <code>ZIP Compression LZMA</code> : compresión LZMA</li><li>• <code>ZIP Compression XZ</code> : compresión XZ</li><li>• <code>ZIP Compression none</code> : sin compresión</li></ul>												
level	Integer	Nivel de compresión. Valores posibles: 1 a 10. Un valor más bajo producirá un archivo más grande, mientras que un valor más alto producirá un archivo más pequeño. Sin embargo, el nivel de compresión influye en el rendimiento. Valores por defecto si se omiten: <ul style="list-style-type: none"><li>• <code>ZIP Compression standard</code> : 6</li><li>• <code>ZIP Compression LZMA</code> : 4</li><li>• <code>ZIP Compression XZ</code> : 4</li></ul>												
encryption	Integer	La encriptación a utilizar si se define una contraseña: <ul style="list-style-type: none"><li>• <code>ZIP Encryption AES128</code> : encriptación AES con una llave de 128 bits.</li><li>• <code>ZIP Encryption AES192</code> : encriptación AES con una llave de 192 bits.</li><li>• <code>ZIP Encryption AES256</code> : encriptación AES con una llave de 256 bits (por defecto si se define la contraseña).</li><li>• <code>ZIP Encryption none</code> : los datos no están encriptados (por defecto si no se define una contraseña)</li></ul>												
contraseña	Texto	Una contraseña a utilizar si se requiere encriptación.												
Histórico	Collection	<ul style="list-style-type: none"><li>• una colección de objetos <code>4D.File</code> o <code>4D.Folder</code> o</li><li>• una colección de objetos con las siguientes propiedades:</li></ul> <table border="1"><thead><tr><th>Propiedad</th><th>Tipo</th><th>Descripción</th></tr></thead><tbody><tr><td>source</td><td>4D.File o 4D.Folder</td><td>File o Folder</td></tr><tr><td>destination</td><td>Texto</td><td>(opcional) - Especifica una ruta de archivo relativa para cambiar la organización del contenido del archivo</td></tr><tr><td>option</td><td>number</td><td>(opcional) - <code>ZIP Ignore invisible files</code> o 0 para comprimir todo el archivo</td></tr></tbody></table>	Propiedad	Tipo	Descripción	source	4D.File o 4D.Folder	File o Folder	destination	Texto	(opcional) - Especifica una ruta de archivo relativa para cambiar la organización del contenido del archivo	option	number	(opcional) - <code>ZIP Ignore invisible files</code> o 0 para comprimir todo el archivo
Propiedad	Tipo	Descripción												
source	4D.File o 4D.Folder	File o Folder												
destination	Texto	(opcional) - Especifica una ruta de archivo relativa para cambiar la organización del contenido del archivo												
option	number	(opcional) - <code>ZIP Ignore invisible files</code> o 0 para comprimir todo el archivo												
retrollamada	4D.Function	Una fórmula de retrollamada que recibirá la progresión de la compresión (0 - 100) en \$1.												

En el parámetro `destinationFile`, pase un objeto `4D.File` describiendo el archivo ZIP a crear (nombre, ubicación, etc.). Se aconseja utilizar la extensión ".zip" si quiere que el archivo ZIP sea procesado automáticamente por cualquier software.

Una vez creado un archivo, puede utilizar el comando [ZIP Read archive](#) para acceder a él.

#### Objeto de estado

El objeto status devuelto contiene las siguientes propiedades:

Propiedad	Tipo	Descripción
statusText	Texto	Mensaje de error (si lo hay): <ul style="list-style-type: none"><li>• No se puede abrir el archivo ZIP</li><li>• No se puede crear un archivo ZIP</li><li>• La contraseña es necesaria para la codificación</li></ul>
status	Integer	Código de estado
success	Booleano	True si el archivo se ha creado con éxito, si no, false

#### Ejemplo 1

Para comprimir un `4D.File`:

```
var $file; $destination : 4D.File
var $status : Object

$destination:=Folder(fk desktop folder).file("MyDocs/file.zip")
$file:=Folder(fk desktop folder).file("MyDocs/text.txt")

$status:=ZIP Create archive($file;$destination)
```

#### Ejemplo 2

Para comprimir un `4D.Folder` sin la carpeta misma:

```
var $folder : 4D.Folder
var $destination : 4D.File
var $status : Object

$destination:=Folder(fk desktop folder).file("MyDocs/Images.zip")
$folder:=Folder(fk desktop folder).folder("MyDocs/Images")

$status:=ZIP Create archive($folder;$destination;ZIP Without enclosing folder)
```

#### Ejemplo 3

Para comprimir una estructura de archivo ZIP con una contraseña y una barra de progreso:

```

var $destination : 4D.File
var $zip;$status : Object
var progID : Integer

$destination:=Folder(fk desktop folder).file("MyDocs/Archive.zip")

$zip:=New object
$zip.files:=Folder(fk desktop folder).folder("MyDocs/Resources").folders()
$zip.password:="password"
$zip.callback:=Formula(myFormulaCompressingMethod($1))

progID:=Progress New //utilizamos el componente 4D Progress

$status:=ZIP Create archive($zip;$destination)

Progress QUIT(progID)

```

myFormulaCompressingMethod :

```

var $1 : Integer
Progress SET PROGRESS(progID;Num($1/100))

```

## Ejemplo 4

Quiere pasar una colección de carpetas y archivos para comprimir al objeto *zipStructure*:

```

var $destination : 4D.File
var $zip;$err : Object
$zip:=New object
$zip.files:=New collection
$zip.files.push(New object("source";Folder(fk desktop folder).file("Tests/text.txt")))
$zip.files.push(New object("source";Folder(fk desktop folder).file("Tests/text2.txt")))
$zip.files.push(New object("source";Folder(fk desktop folder).file("Images/image.png")))

$destination:=Folder(fk desktop folder).file("file.zip")
$err:=ZIP Create archive($zip;$destination)

```

## Ejemplo 5

Desea utilizar un algoritmo de compresión alternativo con un alto nivel de compresión:

```

var $destination : 4D.File
var $zip; $err : Object

$zip:=New object
$zip.files:=New collection
$zip.files.push(Folder(fk desktop folder).folder("images"))
$zip.compression:=ZIP Compression LZMA
$zip.level:=7 //por defecto es 4

$destination:=Folder(fk desktop folder).file("images.zip")
$err:=ZIP Create archive($zip; $destination)

```

## ZIP Read archive

► Histórico

ZIP Read archive ( *zipFile* : 4D.File { ; *password* : Text } ) : 4D.ZipArchive

Parámetros	Tipo		Descripción
zipFile	4D.File	->	Archivos Zip
contraseña	Texto	->	Contraseña del archivo ZIP, si la hay
Resultado	4D.ZipArchive	<-	Objeto archivo

## Descripción

El comando `ZIP Read archive` recupera el contenido del `zipFile` y lo devuelve como un objeto `4D.ZipArchive`.

Este comando no descomprime el archivo ZIP, sólo ofrece una vista de su contenido. Para extraer el contenido de un archivo, es necesario utilizar métodos como `file.copyTo()` o `folder.copyTo()`.

Pase un objeto `4D.File` haciendo referencia al archivo ZIP comprimido en el parámetro `zipFile`. El archivo de destino se abrirá hasta que el `ZIP Read archive` haya terminado de ejecutarse y todos los contenidos/referencias hayan sido extraídos/liberados, entonces se cerrará automáticamente.

Si el `archivo_zip` está protegido por contraseña, es necesario utilizar el parámetro opcional `password` para suministrar una contraseña. Si se requiere una contraseña pero no se pasa al intentar leer el contenido del archivo, se genera un error.

## Objeto archivo

El objeto `4D.ZipArchive` revuelto contiene una sola propiedad, `root`, cuyo valor es un objeto `4D.ZipFolder`. Esta carpeta describe todo el contenido del archivo ZIP.

## Ejemplo

Para recuperar y ver el contenido de un objeto ZIP file:

```
var $archive : 4D.ZipArchive
var $path : 4D.File

$path:=Folder(fk desktop folder).file("MyDocs/Archive.zip")
$archive:=ZIP Read archive($path)
```

Para recuperar la lista de los archivos y carpetas del archivo:

```
$folders:=$archive.root.folders()
$files:=$archive.root.files()
```

Para leer el contenido de un archivo sin extraerlo de la carpeta root:

```
If($files[$i].extension=".txt")
  $txt:=$files[$i].getText()
Else
  $blob:=$files[$i].getContent()
End if
```

Para extraer desde la carpeta root:

```
//extraer un archivo  
$folderResult:=$files[$i].copyTo(Folder(fk desktop folder).folder("MyDocs"))  
  
//extraer todos los archivos  
$folderResult:=$archive.root.copyTo(Folder(fk desktop folder).folder("MyDocs"))
```

## .root

.root : 4D.ZipFolder

### Descripción

La propiedad `.root` contiene una carpeta virtual que permite acceder al contenido del archivo ZIP.

La carpeta `root` y su contenido pueden ser manipulados con las funciones y propiedades [ZipFile](#) y [ZipFolder](#).

Esta propiedad es de sólo lectura.

# ZIPFile

Las siguientes propiedades y funciones de la clase [File](#) están disponibles para los objetos [ZIPFile](#):

APIs disponibles de <a href="#">File API</a> para ZIPFile	Comentario
<code>.copyTo( destinationFolder : 4D.Folder { ; newName : Text } { ; overwrite : Integer } ) : 4D.File</code>	
<code>.creationDate : Date</code>	
<code>.creationTime : Time</code>	
<code>.exists : Boolean</code>	
<code>.extension : Text</code>	
<code>.fullName : Text</code>	
<code>.getContent( ) : 4D.Blob</code>	
<code>.getIcon( { size : Integer } ) : Picture</code>	
<code>.getText( { charSetName : Text { ; breakMode : Integer } } ) : Text</code>	
<code>.getText( { charSetNum : Integer { ; breakMode : Integer } } ) : Text</code>	
<code>.hidden : Boolean</code>	
<code>.isAlias : Boolean</code>	
<code>.isFile : Boolean</code>	
<code>.isFolder : Boolean</code>	
<code>.isWritable : Boolean</code>	Siempre false con archivo ZIP
<code>.modificationDate : Date</code>	
<code>.modificationTime : Time</code>	
<code>.name : Text</code>	
<code>.original : 4D.File</code>	
<code>.original : 4D.Folder</code>	
<code>.parent : 4D.Folder</code>	
<code>.path : Text</code>	Devuelve una ruta relativa al archivo
<code>.platformPath : Text</code>	

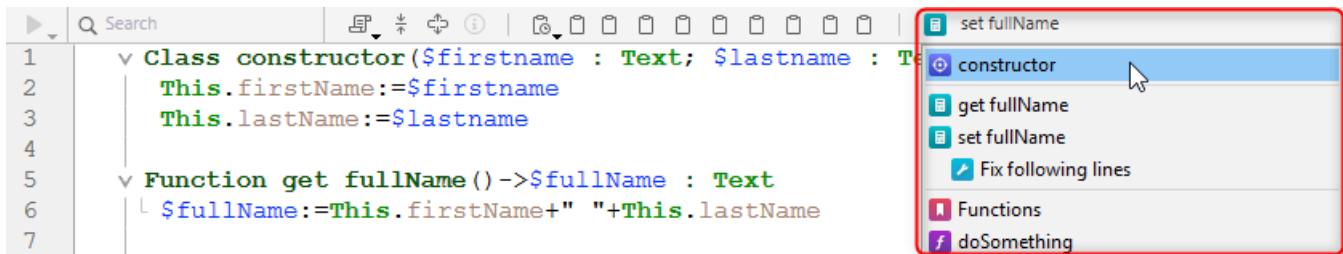
# ZIPFolder

Las siguientes propiedades y funciones de la clase [Folder](#) están disponibles para los objetos [ZIPFolder](#):

APIs disponibles de <a href="#">Folder</a> API para ZIPFolder	Comentario
<code>.copyTo( destinationFolder : 4D.Folder { ; newName : Text } { ; overwrite : Integer } ) : 4D Folder</code>	
<code>.creationDate : Date</code>	La fecha puede ser diferente para la carpeta <code>raíz</code> de una carpeta dentro del archivo
<code>.creationTime : Time</code>	La hora puede ser diferente para la carpeta <code>raíz</code> de una carpeta dentro del archivo
<code>.exists : Boolean</code>	
<code>.extension : Text</code>	
<code>.file( path : Text ) : 4D.File</code>	
<code>.files( { options : Integer } ) : Collection</code>	
<code>.folder( path : Text ) : 4D.Folder</code>	
<code>.folders( { options : Integer } ) : Collection</code>	
<code>.fullName : Text</code>	
<code>.getIcon( { size : Integer } ) : Picture</code>	
<code>.hidden : Boolean</code>	
<code>.isAlias : Boolean</code>	
<code>.isFile : Boolean</code>	
<code>.isFolder : Boolean</code>	
<code>.isPackage : Boolean</code>	
<code>.modificationDate : Date</code>	La fecha puede ser diferente para la carpeta <code>raíz</code> de una carpeta dentro del archivo
<code>.modificationTime : Time</code>	La hora puede ser diferente para la carpeta <code>raíz</code> de una carpeta dentro del archivo
<code>.name : Text</code>	
<code>.original : 4D.Folder</code>	
<code>.parent : 4D.Folder</code>	La carpeta virtual <code>root</code> del archivo no tiene padre. Sin embargo, las carpetas dentro del archivo pueden tener un padre distinto de la raíz.
<code>.path : Text</code>	Devuelve una ruta relativa al archivo
<code>.platformPath : Text</code>	

# Despliegue de navegación

La lista desplegable de navegación le ayuda a organizar su código y a navegar más fácilmente dentro de sus clases y métodos:



Algunas etiquetas se añaden automáticamente y puede complementar la lista desplegable utilizando los [marcadores](#).

## Navegación en el código

Haga clic en un elemento de la lista desplegable para acceder a su primera línea en el código. También puede navegar con las teclas de flecha y presionar Intro.

## Etiquetado automático

Los constructores, las declaraciones de métodos, las funciones y los atributos calculados se etiquetan automáticamente y se añaden a la lista desplegable.

Cuando no hay ninguna etiqueta en la clase/método, la herramienta muestra "Sin etiqueta".

Los siguientes elementos se añaden automáticamente:

Icono	Elemento
∅	Sin etiqueta
🎯	Class constructor o declaración de método
⌚	Atributo calculado (get, set, orderBy y query)
f	Nombre de la función de clase

## Etiquetado manual

Añadiendo marcadores en su código, puede añadir las siguientes etiquetas a la lista desplegable:

Icono	Elemento
🔖	MARK: etiqueta
📝	TODO: etiqueta
🔧	FIXME: tag

Se declaran añadiendo comentarios como:

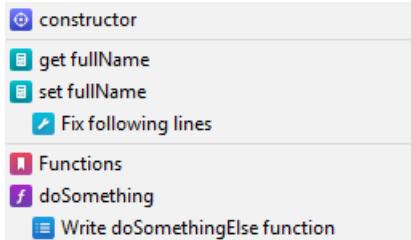
```
// FIXME: Corrige los siguientes elementos
```

Las declaraciones no distinguen entre mayúsculas y minúsculas; escribir `fixme:` tiene el mismo efecto.

Añadir un guión después de la etiqueta `MARK:` dibuja una línea de separación en el editor de código y en el menú desplegable. Así que escribiendo esto:

```
// FIXME: Fix following lines  
  
This.firstName:=Substring($fullName; 1; $p-1)  
This.lastName:=Substring($fullName; $p+1)  
  
//MARK:- Functions  
  
Function doSomething  
  
// TODO: Write doSomethingElse function
```

Esto es lo que resulta:



Todos los marcadores situados dentro de las funciones tienen sangría en la lista desplegable, excepto las etiquetas `MARK:` situadas al final de las funciones y no seguidas de instrucciones. Estos aparecerán en el primer nivel.

## Orden de visualización

Las etiquetas se muestran en su orden de aparición dentro del método/clase.

Para mostrar las etiquetas de un método o de una clase en orden alfabético, realice una de las siguientes acciones:

- haga clic derecho en la herramienta desplegable
- mantenga Cmd en macOS o Alt en Windows, y haga clic en la herramienta de lista desplegable

Las etiquetas dentro de las funciones se mueven con sus elementos padres.

# Básicos

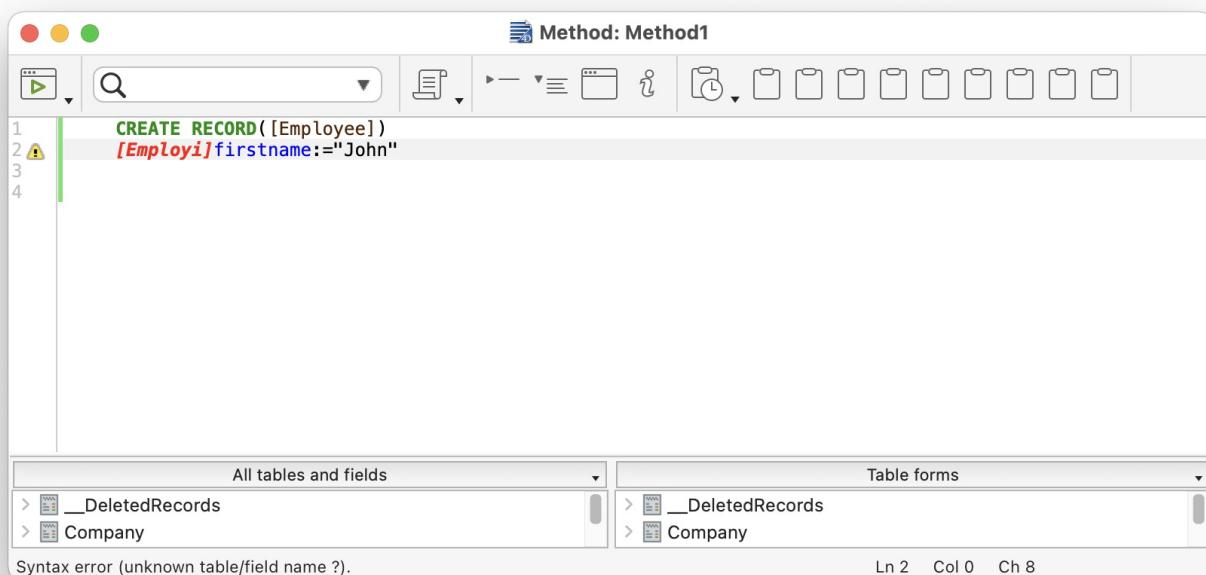
Los errores son comunes. Sería inusual escribir un número importante de líneas de código sin generar errores. Por el contrario, tratar y/o corregir errores también es normal.

El entorno de desarrollo 4D ofrece varias herramientas de depuración para todo tipo de errores.

## Tipos de error

### Errores de digitación

El editor de métodos detecta errores de digitación. Se muestran en rojo y se ofrece información adicional en la parte inferior de la ventana. He aquí un error de escritura:



The screenshot shows the 4D Method Editor window titled "Method: Method1". The code area contains the following lines:

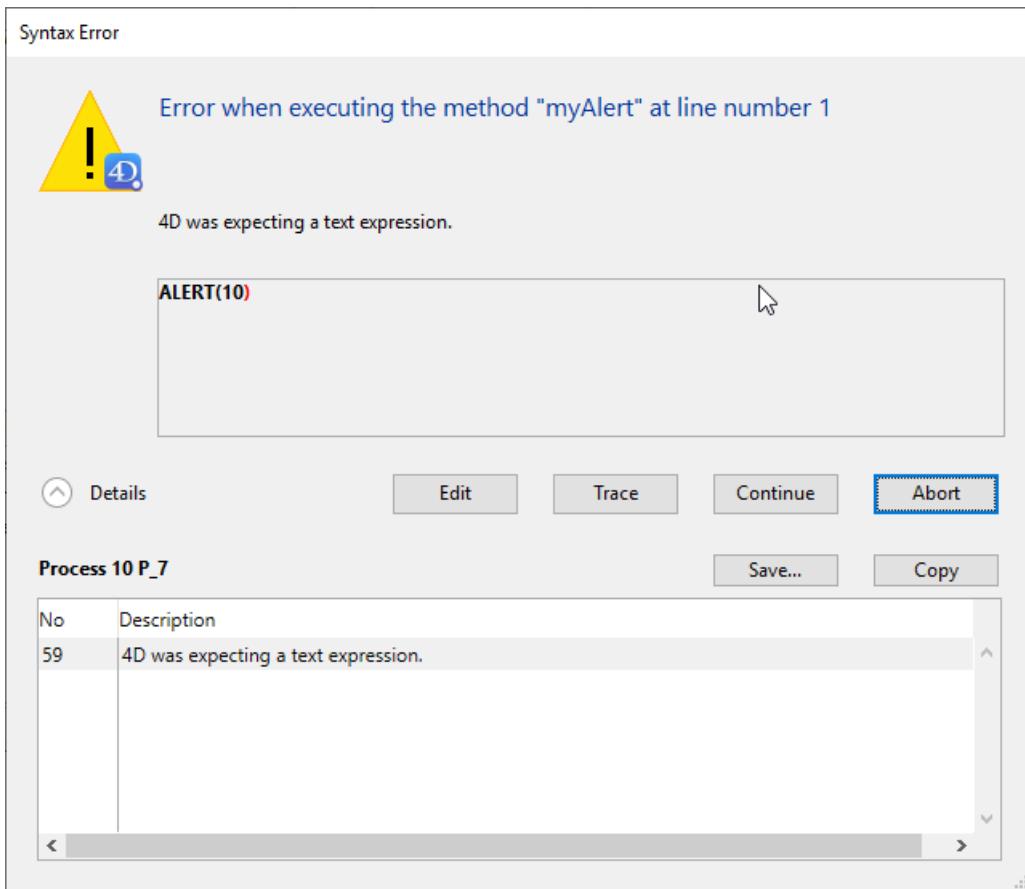
```
1 CREATE RECORD([Employee])
2 [Employi]firstname := "John"
```

Line 2 has a red squiggle under "Employ*i*", indicating a spelling error. The status bar at the bottom left shows the error message: "Syntax error (unknown table/field name ?)." The status bar at the bottom right shows "Ln 2 Col 0 Ch 8".

Estos errores de escritura suelen causar errores de sintaxis (en la imagen anterior, el nombre de la tabla es desconocido). Se obtiene la descripción del error cuando se valida la línea de código. Cuando esto ocurre, corrija el error de digitación y escriba Enter para validar la solución.

### Errores de sintaxis

Algunos errores sólo se pueden capturar cuando se ejecuta el método. La [ventana de error de sintaxis](#) aparece cuando ocurre un error durante la ejecución del código. Por ejemplo:



Expanda el área Detalles para mostrar el último error y su número.

## Errores del entorno

Ocasionalmente, puede que no haya suficiente memoria para crear un BLOB. O, cuando acceda a un documento en el disco, el documento puede no existir o ya estar abierto por otra aplicación. Estos errores en el entorno no se producen directamente por su código o por la forma en que lo escribe. La mayoría de las veces estos errores son fáciles de tratar con un [método de captura de errores](#) instalado utilizando el comando `ON ERR CALL`.

## Errores de diseño o de lógica

Estos son generalmente los tipos de errores más difíciles de encontrar. A excepción de los errores de digitación, todos los tipos de errores listados arriba están cubiertos hasta cierto punto por la expresión "error de diseño o de lógica". Utilice el [depurador](#) para detectarlos. Por ejemplo:

- Puede ocurrir un *error de sintaxis* cuando intenta utilizar una variable que aún no está inicializada.
- Puede ocurrir un *error de entorno* cuando intenta abrir un documento, porque el nombre de ese documento es recibido por una subrutina que no obtuvo el valor correcto como parámetro.

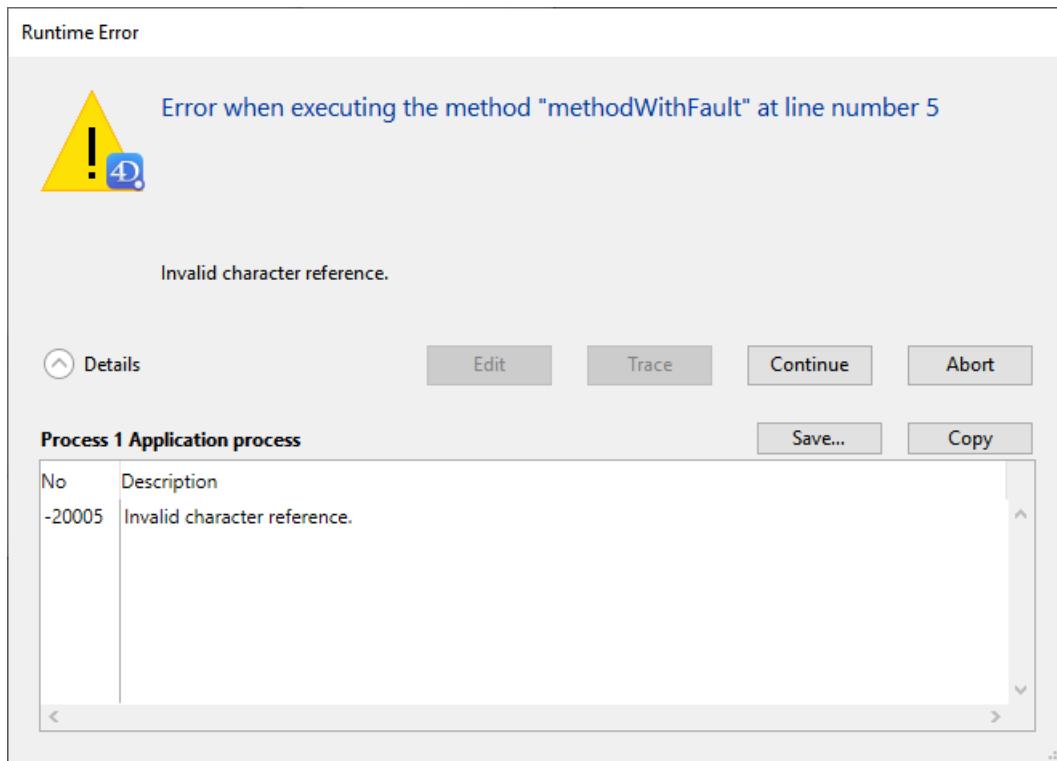
Los errores de diseño o de lógica también incluyen situaciones como:

- Un registro no está correctamente actualizado porque, mientras se llama a `SAVE RECORD`, se olvidó de la primera prueba de si el registro estaba bloqueado o no.
- Un método no hace exactamente lo que espera, ya que la presencia de un parámetro opcional no está probada.

A veces el código que muestra el error puede ser diferente al código que en realidad es el origen del problema.

## Errores de ejecución

En modo Aplicación, puede obtener errores que no ve en modo interpretado. Aquí un ejemplo:

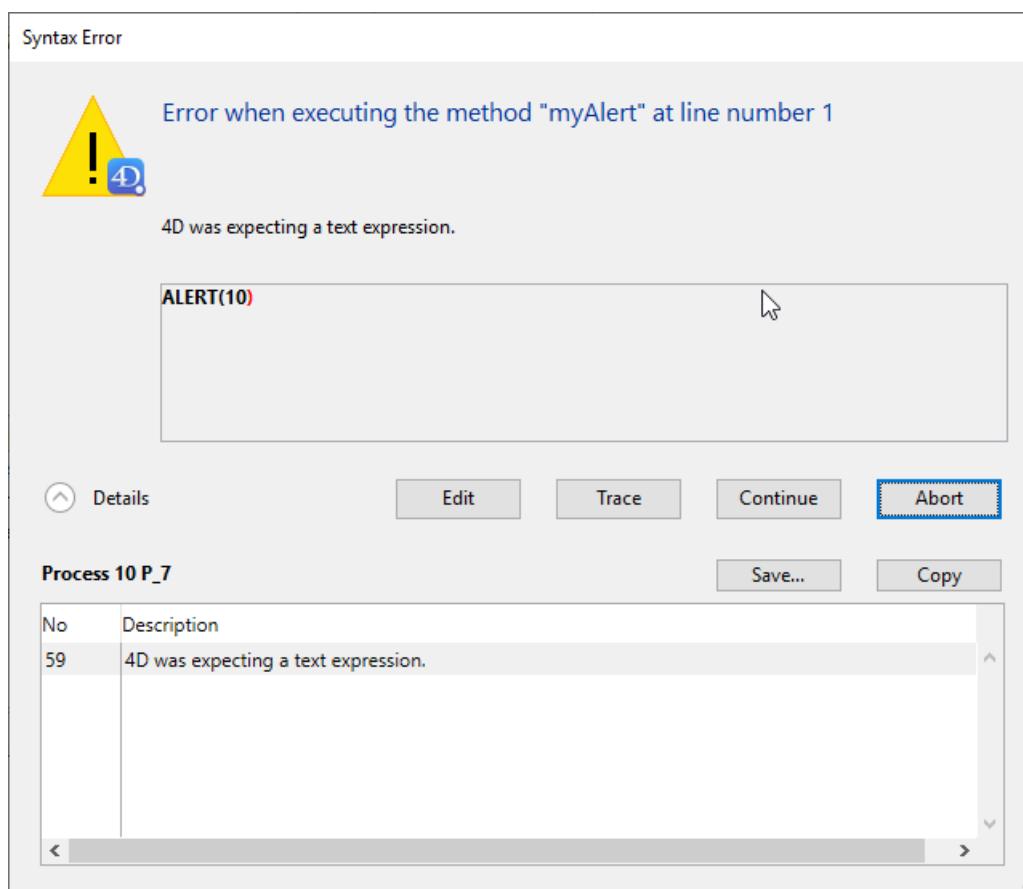


Para encontrar rápidamente el origen del problema, vuelva a abrir la versión interpretada del archivo de estructura, abra el método y vaya a la línea correspondiente.

## Ventana de error de sintaxis

La ventana de error de sintaxis aparece automáticamente cuando se interrumpe la ejecución de un método. Esto puede ocurrir cuando:

- un error impide que el código siga ejecutándose
- el método produce una afirmación falsa (ver el comando `ASSERT`)



El área de texto superior muestra un mensaje describiendo el error. El área de texto inferior muestra la línea que se estaba ejecutando cuando ocurrió el error; el área donde ocurrió el error se resalta. La sección Detalles extendida contiene la "pila" de errores relacionados con el proceso.

La ventana de error de sintaxis propone varias opciones:

- Modificar: detiene la ejecución de todos los métodos. 4D cambia al entorno Diseño y el método con el error se abre en el editor de métodos, permitiéndole solucionarlo. Utilice esta opción cuando reconozca inmediatamente el error y pueda arreglarlo sin más investigación.
- Rastrear: entra en modo Rastrear/Depurador. Se muestra la ventana del [Depurador](#). Si la línea actual solo se ha ejecutado parcialmente, es posible que tenga que hacer clic en el botón rastrear varias veces.
- Continuar: la ejecución continua. La línea con el error puede ser parcialmente ejecutada, dependiendo de donde se encuentre el error. Continúe con precaución: el error puede impedir que el resto de su método se ejecute correctamente. Recomendamos hacer clic en Continuar sólo si el error está en una llamada trivial (como `SET WINDOW TITLE`) que no impide ejecutar y probar el resto de su código.

**Consejo:** para ignorar un error que ocurre repetidamente (por ejemplo, en bucles), puede convertir el botón Continuar en un botón Ignorar. Mantenga presionada la tecla Alt (Windows) u Opción (macOS) y haga clic en el botón Continuar la primera vez que aparece. La etiqueta del botón cambia a Ignorar si el diálogo es llamado de nuevo por el mismo error.

- Abandonar: detiene la ejecución del método y devuelve al estado antes del inicio de la ejecución del método:
  - Si un método formulario o método objeto se está ejecutando en respuesta a un evento, se detiene y se vuelve al formulario.
  - Si el método se está ejecutando desde dentro del entorno de la aplicación, volverá a ese entorno.
- Copiar: copia la información de depuración en el portapapeles. La información describe el entorno interno del error (número, componente interno, etc.). Está formateado como texto tabulado.
- Guardar...: guarda el contenido de la ventana de error de sintaxis y la cadena de llamadas en un archivo `.txt`.

## Depurador

Un error común de principiantes en la detección de errores es hacer clic en `Abandonar` en la ventana de error de sintaxis, volver al editor de métodos, y tratar de averiguar qué está pasando mirando el código. ¡No lo haga! Ahorrará mucho tiempo y energía utilizando siempre el Depurador.

El depurador le permite pasar lentamente por los métodos. Muestra toda la información que necesita para entender por qué ha ocurrido un error. Una vez que tiene esta información, usted sabe cómo arreglar el error.

Otra razón para usar el Depurador es el desarrollo del código. A veces se puede escribir un algoritmo que es más complejo de lo habitual. A pesar de todos los sentimientos de cumplimiento, no puede estar totalmente seguro de que su codificación sea 100% correcta. En lugar de ejecutarlo "ciego", puede utilizar el comando `TRACE` al comienzo de su código, luego ejecutarlo paso a paso para mantener un ojo en lo que sucede.

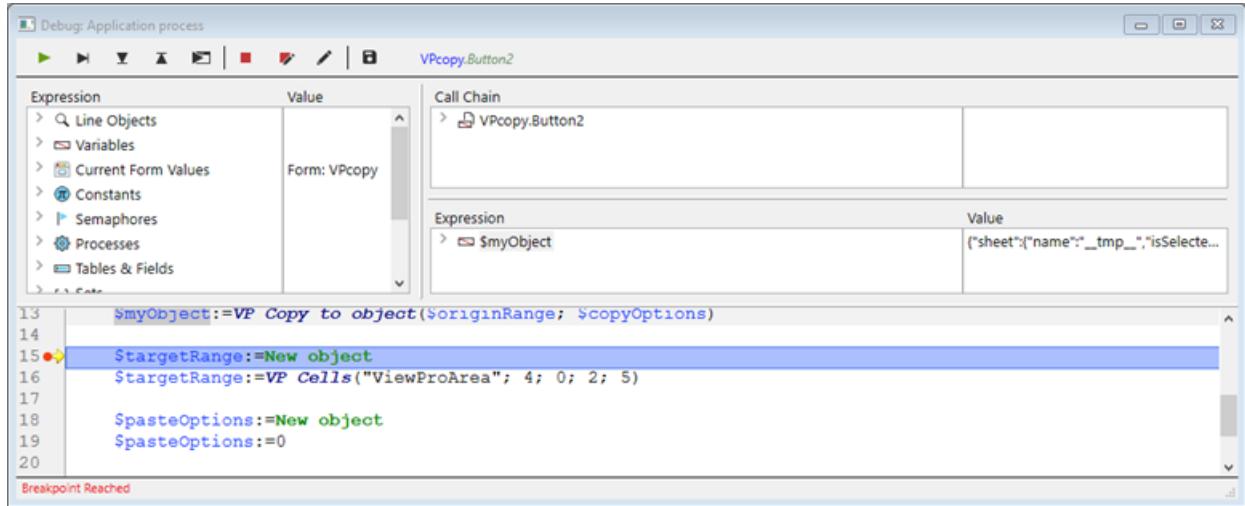
## Rupturas

En el proceso de depuración, puede que necesite saltar el seguimiento de algunas partes del código hasta una línea determinada. O, puede querer rastrear el código cuando una expresión dada tiene un determinado valor (e.. `">$myVar > 1000`), o cada vez que se llama un comando 4D específico.

Estas necesidades están cubiertas por puntos de interrupción y las funciones de captura de comando. Pueden configurarse desde el editor de métodos, el depurador o el Explorador de tiempos de ejecución.

# Depurador

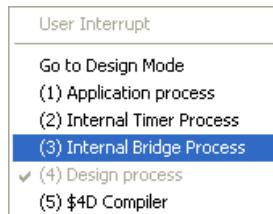
The debugger is useful when you need to spot errors or monitor the execution of methods. It allows you to step through your code slowly and examine the information. This process is called "tracing".



## Llamada al depurador

There are multiple ways to get the Debugger to display:

- Clicking the Trace button in the [Syntax Error window](#)
- Using the [TRACE](#) command
- Clicking the Debug button in the Execute Method window or selecting Run and debug... button in the Method editor
- Using Alt+Shift+Right click (Windows) or Ctrl+Option+Cmd+Click (macOS) while a method is executing, then selecting the process to trace in the pop-up menu:



- Clicking the Trace button when a process is selected in the Process page of the Runtime Explorer.
- Adding a break point in the Method Editor window or in the Break and Catch pages of the Runtime Explorer.

When called, the debugger window provides the name of the method or class function you're currently tracing, and the action causing the initial appearance of the Debugger window. Por ejemplo, en la ventana del depurador arriba:

- *Clients\_BuildLogo* is the method being traced
- The debugger window appeared because it detected a call to the `C_PICTURE` command and this command was one of the commands to be caught

Displaying a new debugger window uses the same configuration as the last window displayed in the same session. If you run several user processes, you can trace them independently and have one debugger window open for each process.

The Debugger window is usually displayed on the machine where the code is executed. With a single-user application, it is always displayed on the machine running the application. Con una aplicación cliente/servidor se muestra:

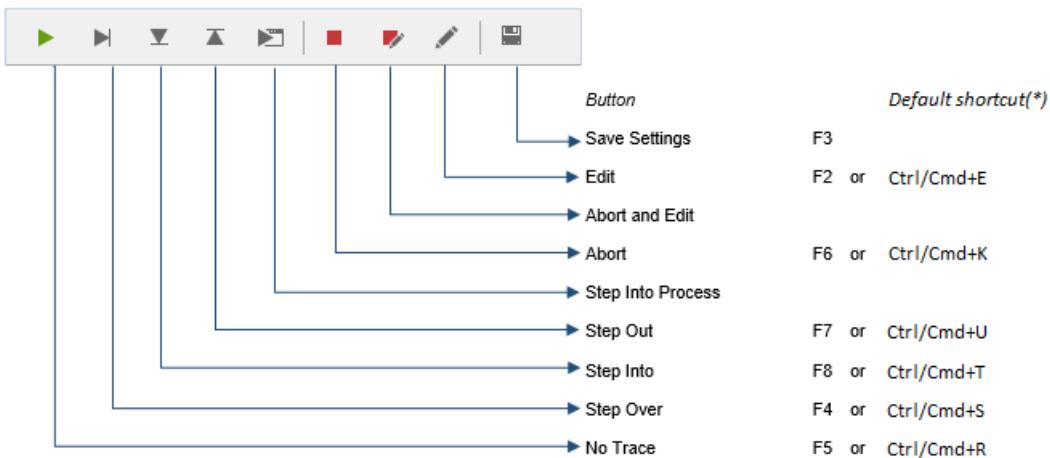
- en el 4D remoto para el código que se ejecuta localmente
- on the server machine for code running on the server (for example, a method with the execute on server option).

If the server is running headless, no debugger window can be displayed on the server, you need to use the

remote debugger. See [Debugging from remote machines](#).

## Botones barra de herramientas

The debugger's tool bar includes several buttons, associated with default shortcuts:



Default shortcuts can be customized in the Shortcuts Page of the Preferences dialog box.

### Fin del rastreo

Detener el modo Seguimiento y reanudar el curso normal de la ejecución del método.

Shift + F5 or Shift + clicking the No Trace button resumes execution. It also disables all the subsequent TRACE calls for the current process.

### Pasar por encima

Executes the current method line, indicated by the program counter (the yellow arrow). El depurador pasa a la siguiente línea.

The Step Over button does not step into subroutines and functions, it stays at the level of the method you're currently tracing. If you want to also trace subroutines and functions calls, use the Step Into button.

In remote debugging, if the method executes on the server, the parent method is called after the last line of the child method executes. If the parent method is executed on the remote side, the Step Over button has the same effect as the No Trace button.

### Paso a paso detallado

When a line that calls another method (subroutine or function) is executed, click this button to display the other method and step through it.

The new method becomes the current (top) method in the [Call Chain Pane](#) of the Debugger window.

When executing a line that does not call another method, this button has the same effect as the Step Over button.

### Abortar

Stops method execution, and returns to the state before the method started executing:

- When tracing a form or object method executing in response to an event: Stops and returns to the form.
- When tracing a method executing from within the Application environment: Stops and returns to the environment.

### Detener y editar

Pausa la ejecución del método. The method that is executing when you click the Abort and Edit button opens in the

## Method Editor.

Tip: Use this button when you know which changes are required in your code, and when these changes are required to pursue the testing of your methods. After you're finished with the changes, rerun the method.

### Editar

Pausa la ejecución del método. The method that is executing at the time you click the Edit button opens in the Method Editor.

If you use this button to modify a method, the modifications are only effective the next time it executes.

Tip: Use this button when you know which changes are required in your code and when they don't interfere with the rest of the code to be executed or traced.

### Parámetros Guardar

Saves the current configuration of the debugger window and makes it the default configuration. Esto incluye:

- el tamaño y la posición de la ventana
- the position of the division lines and the contents of the area that evaluates the expressions

Estos parámetros se almacenan en el proyecto.

This action is not available in remote debugging mode (see [Debugging from Remote Machines](#)).

## Ventana de expresión

The Watch pane is displayed in the top left corner of the Debugger window, below the Execution Control Tool Bar. Aquí un ejemplo:

Expression	Value
▶ 🔎 Line Objects	
◀ 📂 Variables	
▶ 📂 Interprocess	
◀ 📂 Process	
▶ 📂 Document	""
▶ 📂 Error	0
▶ 📂 FldDelimit	9
▶ 📂 OK	0
▶ 📂 RecDelimit	13
▶ 📂 Local	
▶ 📂 Parameters	
▶ 📂 Self	Nil
▶ 📄 Current Form Values	
▶ ⚙ Constants	
▶ 🚧 Semaphores	
▶ 🚧 Processes	
▶ 📊 Tables & Fields	
▶ 📊 Sets	
▶ 📊 Named Selections	
▶ 🌐 Information	
▶ 🌐 Web	

This pane is not available in remote debugging mode.

The Watch Pane displays useful general information about the system, the 4D environment, and the execution environment.

The Expression column displays the names of the objects and expressions. The Value column displays their current corresponding values. Clicking on any value on the right side of the pane allows you to modify the value of the object, if

this is permitted for that object.

At any point, you can drag and drop themes, theme sublists (if any), and theme items to the [Custom Watch Pane](#).

## List of expressions

### Line Objects

This theme allows you to follow the values of objects or expressions:

- used in the line of code to be executed (the one marked with the program counter—the yellow arrow in the [Source Code Pane](#)),
- used in the previous line of code

Since the previous line of code is the one that was just executed before, this theme therefore shows the objects or expressions of the current line before and after that the line was executed. Let's say we run the following method:

```
TRACE
$a:=1
$b:=a+1
$c:=a+b
```

1. A Debugger window opens with the program counter set to the line with `a:=1`. At this point the Line Objects theme displays:

\$a	Indefinido
-----	------------

The variable `$a` has not yet been initialized, but it is shown because it is used in the line that is about to be executed.

2. You click the Step Over button. The program counter is now set to the line `b:=a+1`. At this point, the theme displays:

\$a	1
\$b	Indefinido

The value of the variable `$a` is now 1. The variable `$b` has not yet been initialized, but it is shown because it is used in the line that is about to be executed.

3. You click the Step Over button again. The program counter is now set on the line with `c:=a+b`. At this point the Line Objects theme displays:

\$c	Indefinido
\$a	1
\$b	2

The value of the variable `$b` is now 2. The variable `$c` has not yet been initialized, but it is shown because it is used in the line that is about to be executed.

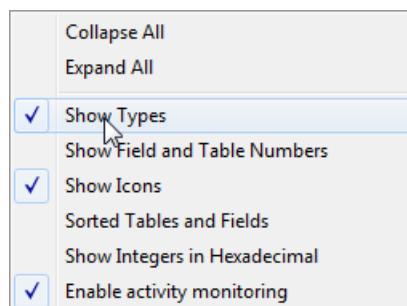
### Variables

This theme consists of the following subthemes:

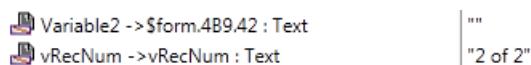
Subtema	Descripción	¿Se pueden modificar los valores?
Interproceso	List of interprocess variables being used at this point	Sí
Proceso	List of process variables used by the current process	Sí
Local	List of local variables used by the method being traced	Sí
Parámetros	Lista de parámetros recibidos por el método	Sí
Self	Pointer to the current object, when tracing an Object Method	No

Arrays, like other variables, appear in the Interprocess, Process, and Local subthemes, depending on their scope. El depurador muestra los primeros 100 elementos. Inside the Value column, you can modify the values of array elements, but not the size of the arrays.

To display the variable types and their internal names, right click and check the Show Types option in the context menu:



Aquí está el resultado:



## Valores actuales del formulario

This theme contains the name of each dynamic object included in the current form, as well as the value of its associated variable:

Current Form Values		Form: debugger
File	bCancel	0
File	bDelete	0
File	Button3	1
File	bValidate	0
File	FirstName	"Tony"
File	ID	"2"
File	List Box1	0 elements
File	List Box1	Listbox sub objects

Some objects, such as list box arrays, can be presented as two distinct objects, the variable of the object itself and its data source.

## Constantes

Like the Constants page of the Explorer window, this theme displays predefined constants provided by 4D. The expressions from this theme cannot be modified.

## Semáforos

This theme lists the local semaphores currently being set. For each semaphore, the Value column provides the name of the process that sets the semaphore. The expressions from this theme cannot be modified. The expressions from this theme cannot be modified.

## Procesos

This theme lists the processes started since the beginning of the working session. The value column displays the time used and the current state for each process (i.e., Executing, Paused, and so on). The expressions from this theme

cannot be modified.

## Tablas y campos

This theme lists the tables and fields in the 4D database. For each Table item, the Value column displays the size of the current selection for the current process as well as the number of locked records.

For each Field item, the Value column displays the value of the field for the current record (except picture and BLOB). You can modify the field values but not the the tables' information.

## Conjuntos

This theme lists the sets defined in the current process (the one you're currently tracing) and the interprocess sets. For each set, the Value column displays the number of records and the table name. The expressions from this theme cannot be modified.

## Selecciones temporales

This theme lists the named selections that are defined in the current process (the one you're currently tracing); it also lists the interprocess named selections. For each named selection, the Value column displays the number of records and the table name. The expressions from this theme cannot be modified.

## Information

This theme contains general information regarding database operation, such as the current default table (if one exists), physical, virtual, free and used memory space, query destination, etc.

## Web

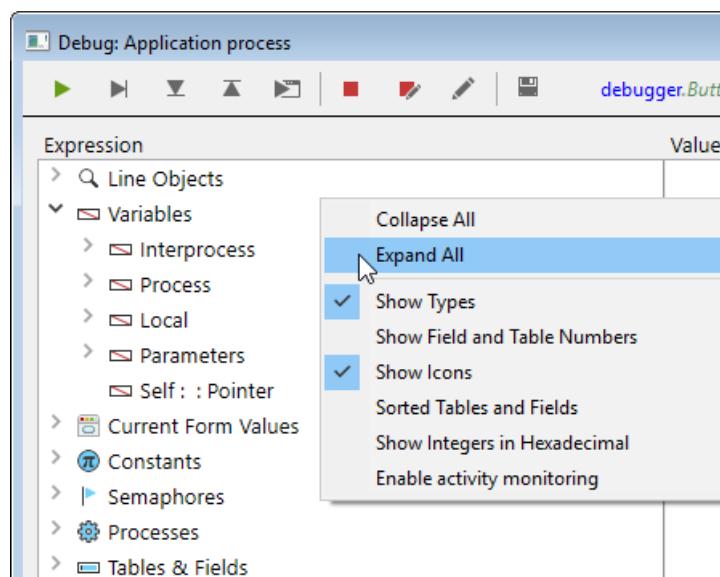
This theme displays information regarding the main Web server of the application (only available if the Web server is active):

- Web File To Send: name of Web file waiting to be sent (if any)
- Web Cache Usage: number of pages present in Web cache as well as its use percentage
- Web Server Elapsed Time: duration of Web server use in hours:minutes:seconds format
- Web Hits Count: total number of HTTP requests received since Web server launch, as well as the instantaneous number of requests per second
- Number of active Web processes: number of active Web processes, all Web processes together

The expressions contained within this theme cannot be modified.

## Menú contextual de la ventana de expresión

Additional options are available from the contextual menu of the Watch pane.



- Collapse All: Collapses all levels of the hierarchical list.
- Expand All: Expand all levels of the hierarchical list.
- Show Types: Displays the type of each item (when appropriate).
- Show Field and Table Numbers: Displays the number of each table or field. Useful if you work with table or field numbers, or with pointers using commands such as `Table` or `Field`.
- Show Icons: Displays an icon denoting the object type for each object. You can turn this option off in order to speed up the display, or just because you prefer to use only the Show Types option.
- Sorted Tables and Fields: Sorts the tables and fields in alphabetical order within their respective lists.
- Show Integers in Hexadecimal: Numbers are usually displayed in decimal notation. Esta opción los muestra en notación hexadecimal. Note: To enter a numeric value in hexadecimal, type `0x` (zero + "x"), followed by the hexadecimal digits.
- Enable activity monitoring: Activates the monitoring of activity (advanced checking of internal activity of the application) and displays the information retrieved in the additional themes: Scheduler, Web and Network.

## Panel de la cadena de llamadas

A method may call other methods or class functions, which may call other methods or functions. The Call Chain pane lets you keep track of that hierarchy.

Method / Parameter	Type
\$0	Undefined
\$1	Undefined -> [Employee]
\$2	Undefined -> [Employee]ID
\$3	Z

Each main level item is the name of a method or class function. The top item is the one you are currently tracing, the next main level item is the name of the caller (the method or function that called the one you are currently tracing), the next one is the caller's caller, and so on.

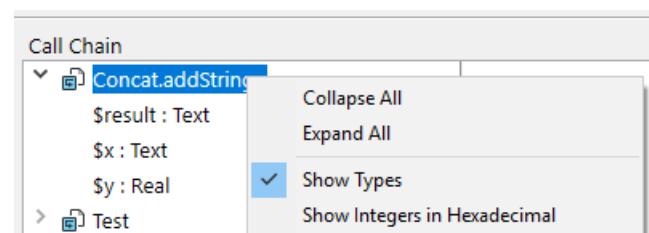
En la imagen de arriba:

- `thirdMethod` has not received any parameter
- `$0` is currently undefined, as the method did not assign any value to `$0` (because it has not executed this assignment yet or because the method is a subroutine and not a function)
- `secondMethod` has received three parameters from `firstMethod` :
  - `$1` is a pointer to the `[Employee]` table
  - `$2` is a pointer to the `ID` field in the `[Employee]` table
  - `$3` es un parámetro alfanumérico cuyo valor es "Z"

You can double-click the name of any method to display its contents in the [Source Code Pane](#).

Clicking the icon next to a method or function name expands or collapses the parameters and the result (if any). Los valores aparecen en el lado derecho del panel. Clicking on any value on the right side allows you to change the value of any parameter or function result.

To display the parameter type, check the Show types option in the contextual menu:



After you deploy the list of parameters, you can drag and drop parameters and function results to the [Custom Watch Pane](#).

You can also use the [Get call chain](#) command to retrieve the call chain programmatically.

## Panel de vigilancia personalizado

The Custom Watch Pane is useful for evaluating expressions. It is similar to the [Watch Pane](#), except here you decide which expressions are displayed. Todo tipo de expresión puede ser evaluada:

- campo
- variable
- puntero
- cálculo
- Comando 4D
- method
- y cualquier otra cosa que devuelva un valor

Expression	Value
\$text	"Hello, World!"
\$calcResult	3
\$pField	->[Employee]ID
\$myBlob	10 Ko

You can evaluate any expression that can be shown in text form. This does not cover picture and BLOB fields or variables. To display BLOB contents, you can use BLOB commands, such as [BLOB to text](#).

## Gestión de expresiones

There are several ways to add expressions to the list:

- Drag and drop an object or expression from the Watch Pane or the Call Chain Pane
- Select an expression in the [Source Code pane](#) and press **ctrl+D** (Windows) or **cmd+D** (macOS)
- Double-click somewhere in the empty space of the Custom Watch Pane (adds an expression with a placeholder name that you can edit)

Puede introducir cualquier fórmula que devuelva un resultado.

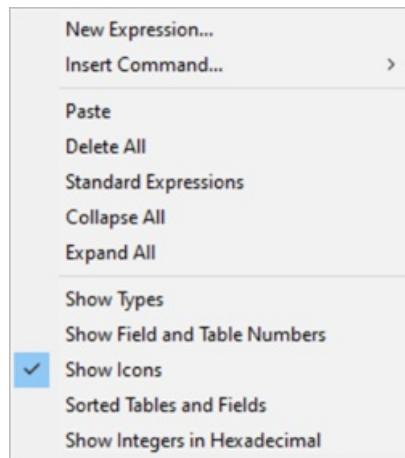
To edit an expression, click on it to select it, then click again or press **Enter** on your keyboard.

To delete an expression, click on it to select it, then press **Backspace** or **Delete** on your keyboard.

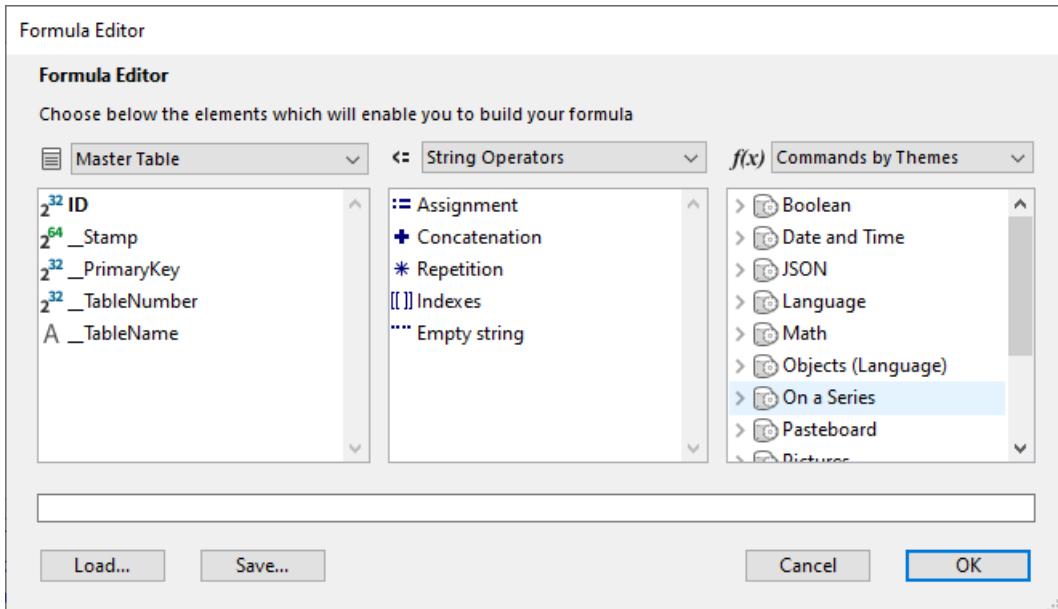
**Warning:** Be careful when you evaluate a 4D expression modifying the value of one of the System Variables (for instance, the **OK** variable) because the execution of the rest of the method may be altered.

## Menú contextual de la ventana de expresión

The Custom Watch Pane's context menu gives you access the 4D formula editor and other options:



New Expression : This inserts a new expression and displays the 4D Formula Editor.



For more information on the Formula Editor, see the [4D Design Reference manual](#).

- Insert Command: Shortcut for inserting a 4D command as a new expression.
- Delete All: Removes all expressions from the Custom Watch Pane.
- Standard Expressions: Copies the Watch Pane's list of expressions.

This option is not available in remote debugging mode (see [Debugging from Remote Machines](#)).

- Collapse All/Expand All: Collapses or Expands all the hierarchical lists.
- Show Types: Displays the type of each item in the list (when appropriate).
- Show Field and Table Numbers : Displays the number of each table or field of the Fields. Useful if you work with tables, field numbers or pointers using the commands such as `Table` or `Field` .
- Show Icons: Displays an icon denoting the type of each item.
- Sorted Tables and Fields: Displays the table and fields in alphabetical order.
- Show Integers in Hexadecimal: Displays numbers using hexadecimal notation. To enter a numeric value in hexadecimal, type 0x (zero + "x"), followed by the hexadecimal digits.

## Panel de código fuente

The Source Code Pane shows the source code of the method or function currently being traced.

This area also allows you to add or remove [break points](#).

## Tips

Hover your pointer over any expression to display a tool tip that indicates:

- el tipo declarado de la expresión
- el valor actual de la expresión

```
1 // $1 contains the primary key of the manager
2 // $2 is a pointer to the resulting array
3
4 QUERY([Employee];[Employee]ID;=$1) // finds the employee whose primary key w
5 APPEND TO ARRAY($2->,[Employee]firstname+" "+[Employee]lastname) // and adds
6
7 QUERY([Employee];[Employee]managerID;=$1,[Employee]ID) // now finds all direct
8
9 If (Records in selection([Employee])>0) // if there are some
10
11 C_LONGINT($i)
12 ARRAY LONGINT($IDS;0)
13 SELECTION TO ARRAY([Employee]ID;$IDS)
```

Esto también funciona con las selecciones:

```
1 // $1 contains the primary key of the manager
2 // $2 is a pointer to the resulting array
3
4 QUERY([Employee];[Employee]ID;=$1) // finds the employee whose primary key w
5 APPEND TO ARRAY($2->,[Employee]firstname+" "+[Employee]lastname) // and adds
6
7 QUERY([Employee];[Employee]managerID;=$1,[Employee]ID) // now finds all direct
8
9 If (Records in selection([Employee])>0) // if there are some
10
11 C_LONGINT($i)
12 ARRAY LONGINT($IDS;0)
13 SELECTION TO ARRAY([Employee]ID;$IDS)
```

## Añadir expresiones al panel de control personalizado

You can copy any selected expression from the Source Code Pane to the [Custom Watch Pane](#).

1. In the Source code pane, select the expression to evaluate
2. Haga una de las siguientes cosas:
  - Drag and drop the selected text to the Expression area of the Custom Watch Pane
  - Press Ctrl+D (Windows) or Cmd+D (macOS)
  - Right-click the selected text > Copy to Expression Pane

## Program Counter

The yellow arrow in the left margin of the Source Code pane is called the program counter. Marca la siguiente línea a ejecutar.

By default, the program counter line (also called the running line) is highlighted in the debugger. You can customize the highlight color in the [Methods page of the Preferences](#).

### Mover el contador del programa

For debugging purposes, you can move the program counter for the method at the top of the call chain (the method currently executing). To do so, click and drag the yellow arrow to another line.

This only tells the debugger to pursue tracing or executing from a different point. It does not execute lines or cancel their execution. All current settings, fields, variables, etc. are not impacted.

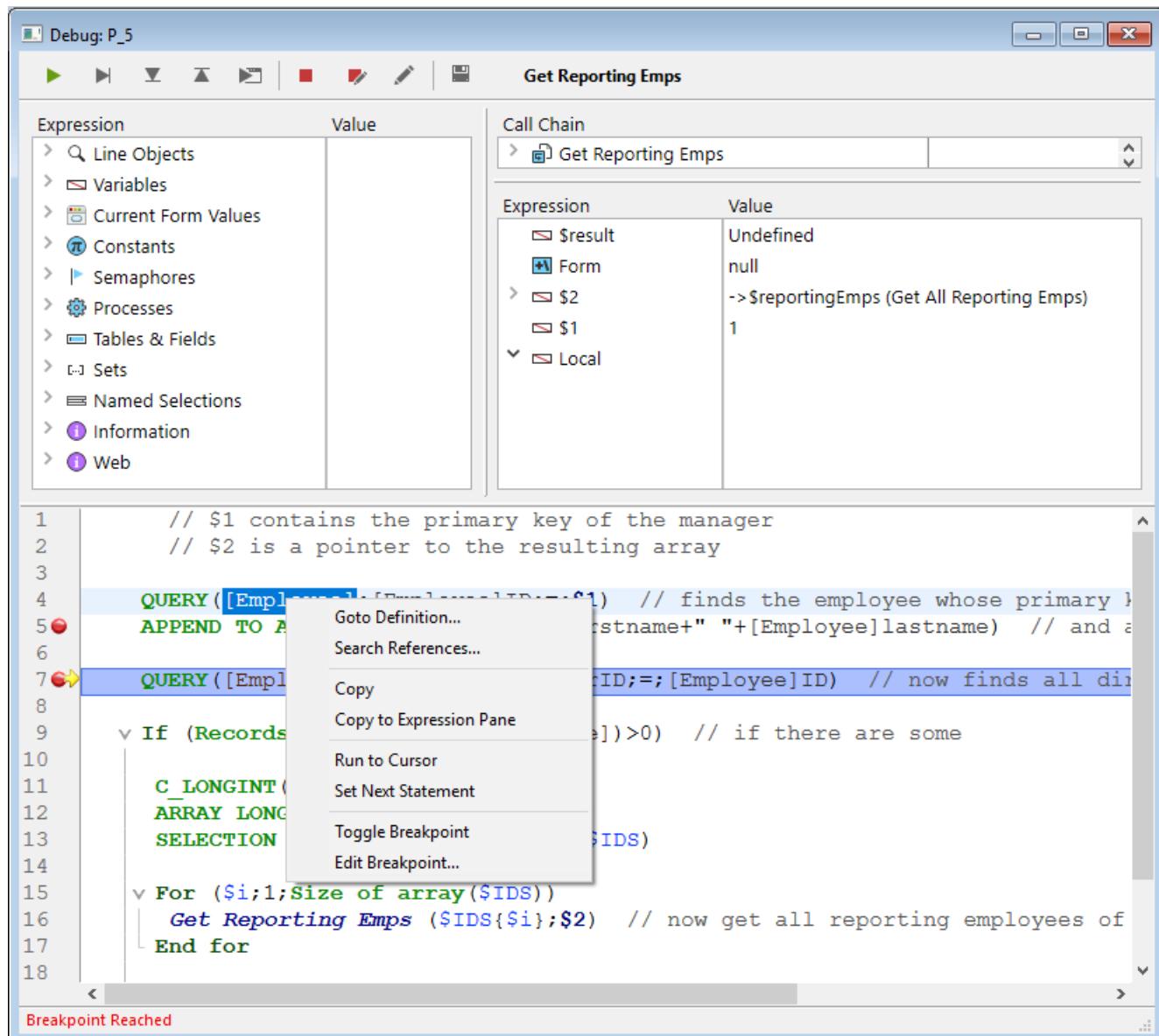
Por ejemplo:

```
// ...
If(This condition)
    DO_SOMETHING
Else
    DO_SOMETHING_ELSE
End if
// ...
```

Say the program counter is set to the line `If (This condition)`. When you click the Step over button, the program counter moves directly to the `DO_SOMETHING_ELSE` line. To examine the results of the `DO_SOMETHING` line, you can move the program counter to that line and execute it.

## Menú contextual

The contextual menu of the Source Code Pane provides access to several functions that are useful when executing methods in Trace mode:



- **Goto Definition:** Goes to where the selected object is defined. Este comando está disponible para:
  - *Project methods:* displays method contents in a new window of the Method editor
  - *Fields:* Displays field properties in the inspector of the Structure window
  - *Tables:* Displays table properties in the inspector of the Structure window
  - *Forms:* Displays form in the Form editor
  - *Variables* (local, process, interprocess or \$n parameter): displays the line in the current method or among the compiler methods where the variable is declared

- Search References (also available in Method editor): Searches all project objects (methods and forms) in which the current element of the method is referenced. The current element is the one selected or the one where the cursor is located. This can be the name of a field, variable, command, string, and so on. Search results are displayed in a new standard results window.
- Copy: Standard copy of the selected expression to the pasteboard.
- Copy to Expression Pane : Copy the selected expression to the Custom Watch Pane.
- Run to Cursor: Executes statements found between the program counter and the selected line of the method (where the cursor is found).
- Set Next Statement: Moves program counter to the selected line without executing this line or any intermediate ones. The designated line is only run if the user clicks on one of the execution buttons.
- Toggle Breakpoint (also available in Method editor): Alternately inserts or removes the breakpoint corresponding to the selected line. This modifies the breakpoint permanently: for instance, if you remove a breakpoint in the debugger, it no longer appears in the original method.
- Edit Breakpoint (also available in Method editor): Displays the Breakpoint Properties dialog box. Any changes made modify the breakpoint permanently.

## Buscar siguiente/anterior

Specific shortcuts allow you to find strings identical to the one selected:

- To search for the next identical strings, press Ctrl+E (Windows) or Cmd+E (macOS)
- To search for the previous identical strings, press Ctrl+Shift+E (Windows) or Cmd+Shift+E (macOS)

The search is carried out only if you select at least one character in the Source code pane.

## Atajos

This section lists all the shortcuts available in the debugger window.

The tool bar also has [shortcuts](#).

### Ventana de evaluación & Subventana de evaluación personalizada

- Double-click an item in the Watch Pane to copy it to the Custom Watch Pane
- Double-Click in the Custom Watch Pane to create a new expression

### Panel de código fuente

- Click in the left margin to set or remove break points.
- Alt+Shift+Click (Windows) or Option+Shift+Click (macOS) sets a temporary break point.
- Alt-Click (Windows) or Option-Click displays the Edit Break window for a new or existing break point.
- A selected expression or object can be copied to the Custom Watch Pane by simple drag and drop.
- Ctrl+D (Windows) or Cmd+D (macOS) key combinations copy the selected text to the Custom Watch Pane.
- Ctrl+E (Windows) or Cmd+E (macOS) key combinations find the next strings identical to the one selected.
- Ctrl+Shift+E (Windows) or Cmd+Shift+E (macOS) key combinations find the previous strings identical to the one selected.

### Todas las ventanas

- Ctrl + +/- (Windows) or Command + +/- (macOS) increases or decreases the font size for a better readability. The modified font size is also applied to the Method editor and is stored in the Preferences.
- Ctrl + \* (Windows) or Command + \* (macOS) forces the updating of the Watch Pane.
- When no item is selected in any pane, press Enter to step over.
- When an item value is selected, use the arrows keys to navigate through the list.
- When editing an item, use the arrow keys to move the cursor. Use Ctrl-A/X/C/V (Windows) or Command-A/X/C/V (macOS) as shortcuts to the Select All/Cut/Copy/Paste menu commands of the Edit menu.



# Puntos de ruptura y captura de comandos

## Generalidades

Breakpoints and command catching are very efficient debugging techniques. Both have the same effect: they pause the code execution (and display the debugger window if not already displayed) at a desired step.

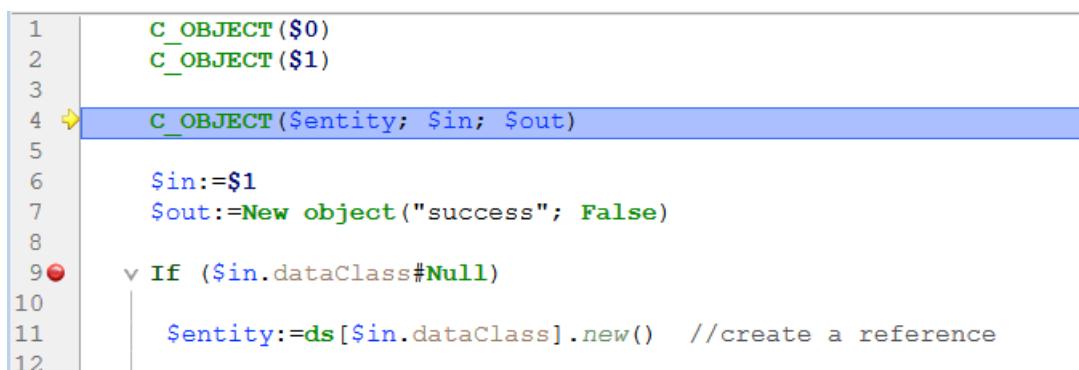
You set breakpoints on any line of code where you want the execution to be paused. You can associate a condition to the break point.

Catching a command enables you to start tracing the execution of any process as soon as a command is called by that process.

## Breakpoints

To create a break point, click in the left margin of the Source Code pane in the debugger or in the Method editor.

In the following example, a break point (the red bullet) has been set, in the debugger, on the line `If ($in.dataClass#Null)` :



The screenshot shows a code editor with the following lines of code:

```
1 C_OBJECT($0)
2 C_OBJECT($1)
3
4 C_OBJECT($entity; $in; $out)
5
6 $in:=$1
7 $out:=New object("success"; False)
8
9 If ($in.dataClass#Null)
10
11     $entity:=ds[$in.dataClass].new() //create a reference
12
```

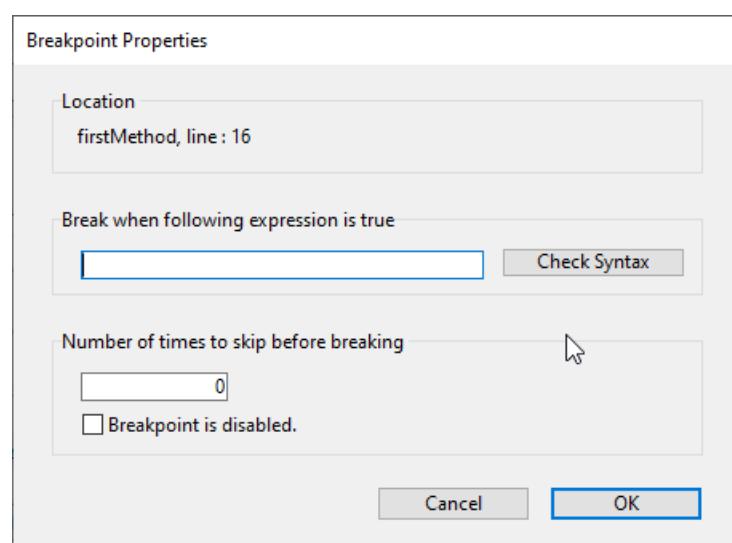
A red circular breakpoint icon is positioned next to the opening brace of the `if` statement on line 9. The line number 9 is also highlighted with a red circle.

In the above example, clicking the `No Trace` button resumes normal execution up to the line marked with the break point. That line is not executed itself — you are taken back to trace mode. Setting a break point beyond the program counter and clicking the `No Trace` button allows you to skip portions of the method being traced.

To remove a break point, click the corresponding bullet.

## Propiedades de los puntos de interrupción

You can edit the behavior of a breakpoint using the Breakpoint Properties window:



This window is available from the Method Editor or the [Source Code Pane](#). Puede:

- right-click a line and select `Edit Breakpoint` in the contextual menu, or
- `Alt+click` (Windows) or `Option+click` (macOS) in the left margin.

If a break point already exists, the window is displayed for that break point. Otherwise, a break point is created and the window is displayed for the newly created break point.

A continuación se describen las propiedades:

- Location: indicates the name of the method and the line number attached to the breakpoint.
- Break when following expression is true : You can create conditional breakpoints by entering a 4D formula that returns `True` or `False`. For example, insert `Records in selection(\[aTable]\)=0` to make sure the break occurs only if there no record selected for the table `[aTable]`. Breakpoint conditions are available in the Condition column of the [Break list](#).
- Number of times to skip before breaking : You can attach a breakpoint to a line located in a loop structure (While, Repeat, or For) or located in subroutine or function called from within a loop.
- Breakpoint is disabled: If you currently do not need a break point, but might need it later, you can temporarily disable it. A disabled break point appears as a dash (-) instead of a bullet (•)|

## Breakpoints in remote debugging

La lista de puntos de interrupción se almacena localmente. In remote debugging mode, if the attached debugger is a remote 4D, the remote break point list replaces temporarily the server break point list during the debugging session.

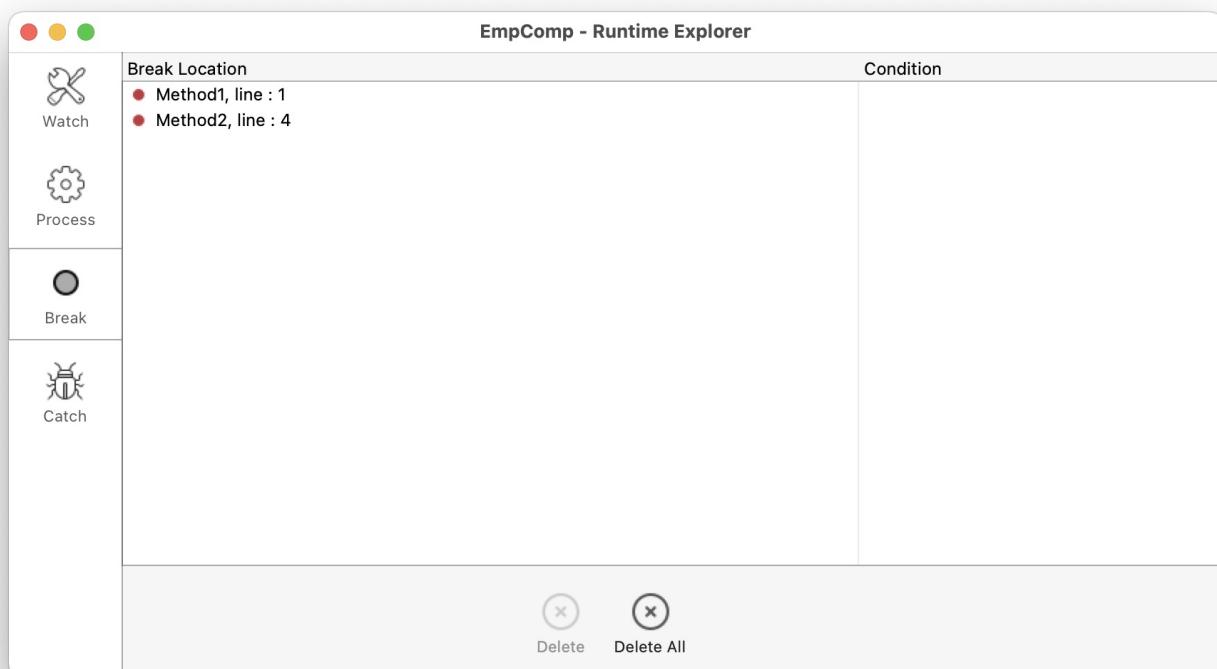
The server break point list is automatically restored if it becomes again the attached debugger.

## Lista de rupturas

The Break list is a page of the Runtime Explorer that lets you manage the breakpoints created in the Debugger Window or in the Method Editor. For more information on the Runtime Explorer, see its dedicated page in [the Design reference manual](#).

Para abrir la página de la lista de puntos de ruptura:

1. From the Run menu, click Runtime Explorer...
2. Click the Break tab to display the Break list:



Utilizando esta ventana, puede:

- Set conditions for breakpoints in the Conditions column
- Enable or disable breakpoints by clicking the bullets in the margin. Los puntos de interrupción desactivados muestran balas transparentes
- Delete breakpoints by pressing the `Delete` or `Backspace` key, or click on the Delete button below the list.
- Open the methods where the breakpoint are located by double-clicking any line in the list

No puede añadir nuevos puntos de interrupción desde esta ventana. Breakpoints can only be created from within the Debugger window or the Method Editor.

## Comandos de captura

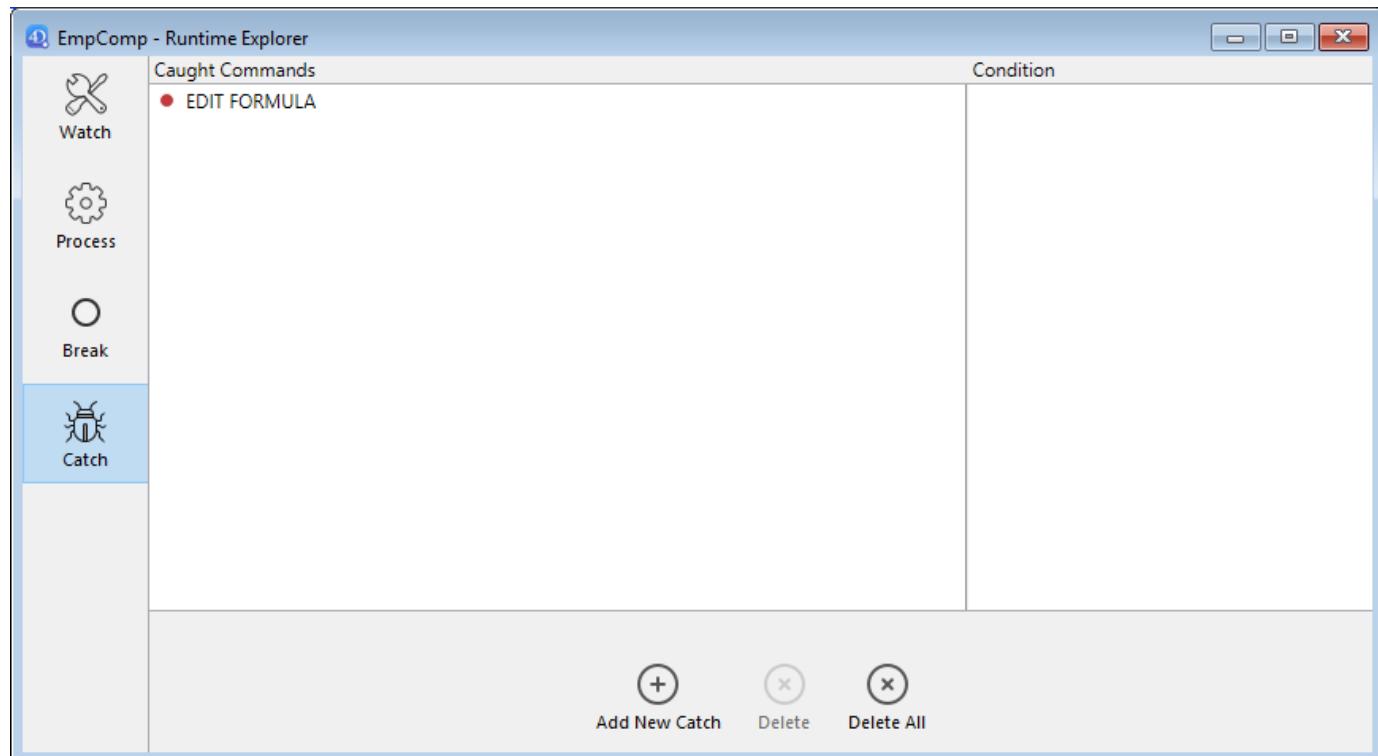
The Catch tab of the Runtime Explorer lets you add additional breaks to your code by catching calls to 4D commands. Unlike a break point, which is located in a particular project method (and therefore triggers a trace exception only when it is reached), the scope of catching a command includes all the processes that execute 4D code and call that command.

Catching a command is a convenient way to trace large portions of code without setting break points at arbitrary locations. For example, if a record that should not be deleted is deleted after you've executed one or several processes, you can try to reduce the field of your investigation by catching commands such as `DELETE RECORD` and `DELETE SELECTION`. Each time these commands are called, you can check if the record in question has been deleted, and thus isolate the faulty part of the code.

Feel free to combine breakpoints and command catching.

To open the Caught Commands page:

1. Choose Run > Runtime explorer... to open the Runtime Explorer.
2. Click Catch to display the Caught Commands List:



This page lists the commands to be caught during execution. Se compone de dos columnas:

- The left column displays the Enable/Disable status of the caught command, followed by the name of the command
- The right column displays the condition associated with the caught command, if any

Para añadir un punto de interrupción en el comando:

1. Click on the Add New Catch button (in the shape of a +) located below the list. A new entry is added to the list with the `ALERT` command as default

2. Click the ALERT label, type the name of the command you want to catch, then press Enter.

To enable or disable a caught command, click on the bullet (•) in front of the command label. The bullet is transparent when the command is disabled.

Disabling a caught command has almost the same effect as deleting it. During execution, the debugger spends almost no time on the entry. The advantage of disabling an entry is that you do not have to recreate it when you need it again.

Para eliminar un punto de interrupción en el comando:

1. Seleccione un comando en la lista.
2. Press Backspace or Delete on your keyboard or click on the Delete button beneath the list (Delete All removes all commands in the list).

## Setting a Condition for catching a command

1. Haga clic en la entrada en la columna derecha
2. Enter a 4D formula (expression, command call or project method) that returns a Boolean value.

Para eliminar una condición, borre su fórmula.

Adding conditions allows you to stop execution when the command is invoked only if the condition is met. For example, if you associate the condition `Records in selection(\[Emp]>10)` with the break point on the `DELETE SELECTION` command, the code will not be stopped during execution of the `DELETE SELECTION` command if the current selection of the [Emp] table only contains 9 records (or less).

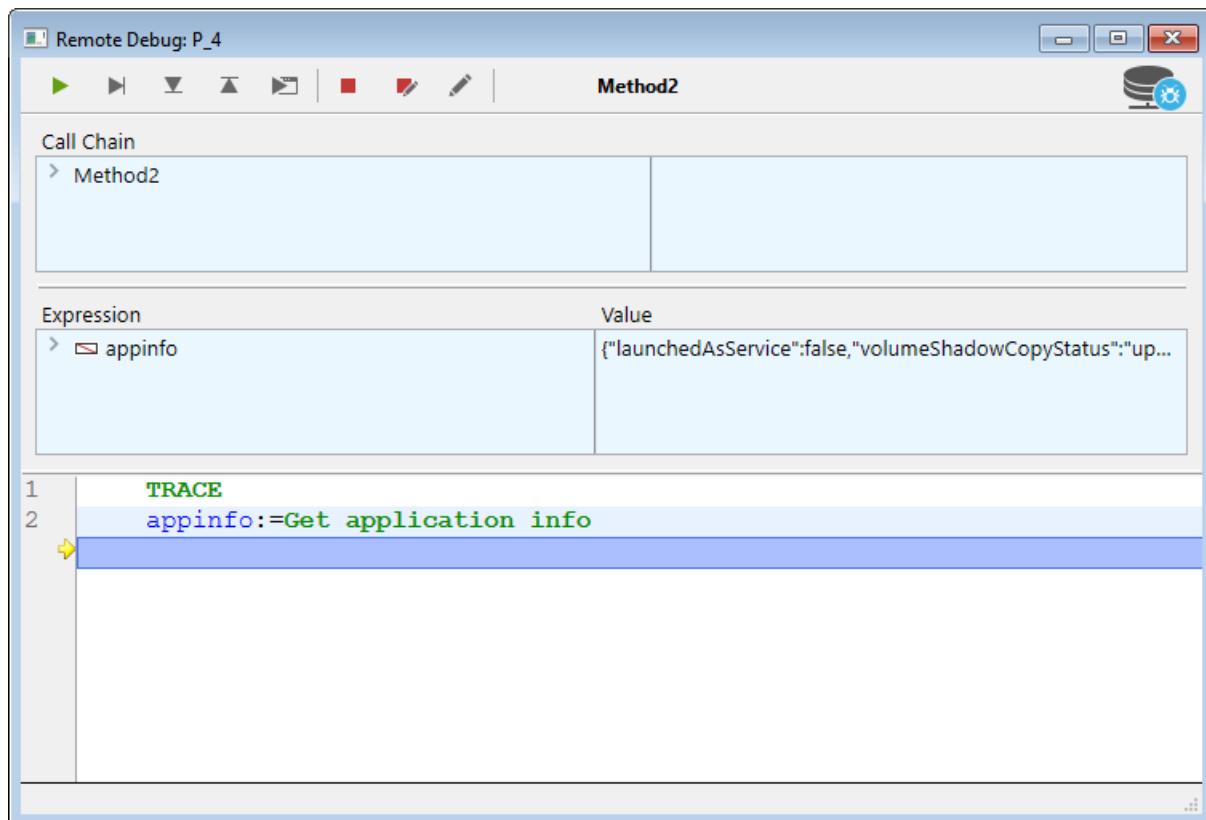
Adding conditions to caught commands slows the execution, because the condition has to be evaluated each time an exception is met. On the other hand, adding conditions accelerates the debugging process, because 4D automatically skips occurrences that do not match the conditions.

# Depuración desde máquinas remotas

## Generalidades

Cuando una base de datos 4D se ejecuta en 4D Server en modo interpretado, puede depurar el código 4D que se ejecuta en el servidor desde un cliente 4D remoto conectado al proyecto. Sólo tiene que adjuntar el depurador a una máquina remota específica y la ejecución del código puede ser monitoreada en el depurador directamente en la máquina remota.

En una máquina remota, la [ventana de depuración](#) muestra un ícono de servidor específico y un color de fondo azul para indicar que está depurando el código servidor:



Esta funcionalidad es especialmente útil cuando 4D Server se ejecuta en modo sin interfaz (ver [Command Line Interface](#)), o cuando el acceso a la máquina del servidor no es fácil.

## Depuradores adjuntos

Sólo un depurador puede depurar una aplicación 4D Server en un momento dado. Se llama el depurador asociado. El depurador asociado puede ser:

- el depurador local de 4D Server (por defecto) - si el servidor no está ejecutando sin interfaz.
- el depurador de un cliente 4D remoto - si la sesión remota tiene acceso al modo Diseño.

El depurador asociado es llamado cada vez que se encuentra un 4D Server:

- un punto de ruptura
- un comando `TRACE`
- un comando de captura
- un error

Tenga en cuenta que los mensajes de error se envían a la máquina depuradora asociada. Esto significa que en el caso de un depurador remoto, los mensajes de error del servidor se muestran en el cliente 4D remoto.

Note que:

- El código ejecutado en el método `On Server Startup Database` no se puede depurar de forma remota. Sólo se puede depurar del lado del servidor
- Si no hay un depurador asociado, el código en ejecución no se detiene con los comandos de depuración

## Asociar al depurador

Por defecto, cuando se inicia una aplicación interpretada:

- si 4D Server no se está ejecutando sin interfaz, el depurador está conectado al servidor,
- si 4D Server se ejecuta sin interfaz, no se asocia ningún depurador.

Puede asociar el depurador a cualquier cliente 4D remoto que pueda conectarse a la aplicación 4D Server.

La sesión usuario del cliente 4D remoto debe tener acceso al entorno de diseño de la base de datos.

Para asociar el depurador a un cliente 4D remoto:

1. En la barra de menús de 4D Server, seleccione Editar > Separar depurador para que el depurador esté disponible para las máquinas remotas (este paso es inútil si 4D Server está funcionando sin interfaz).
2. En un cliente 4D remoto conectado al servidor, seleccione Ejecutar > Adjuntar depurador remoto

Si se acepta el archivo adjunto (ver [Solicitudes de archivo adjunto rechazadas](#)), el comando de menú se convierte en Desconectar depurador remoto.

El depurador se conecta entonces al cliente 4D remoto:

- hasta el final de la sesión usuario
- hasta que se seleccione Detach Remote Debugger

Para volver a conectar el depurador al servidor:

1. En el cliente 4D remoto que tiene el depurador conectado, seleccione Ejecutar > Separar depurador remoto.
2. En la barra de menú de 4D Server, seleccione Editar > Adjuntar depurador.

When the debugger is attached to the server (default), all server processes are automatically executed in cooperative mode to enable debugging. This can have a significant impact on performance. When you don't need to debug on the server machine, it is recommended to detach the debugger and attach it to a remote machine if necessary.

## Adjuntar el depurador al inicio

4D allows you to automatically attach the debugger to a remote 4D client or the server at startup:

- On the server (if not headless), this option is named Attach Debugger At Startup. When the server is started, it automatically attaches the debugger (default).

Warning: If this option is selected for a server which is subsequently launched in headless mode, the debugger won't be available for this server.

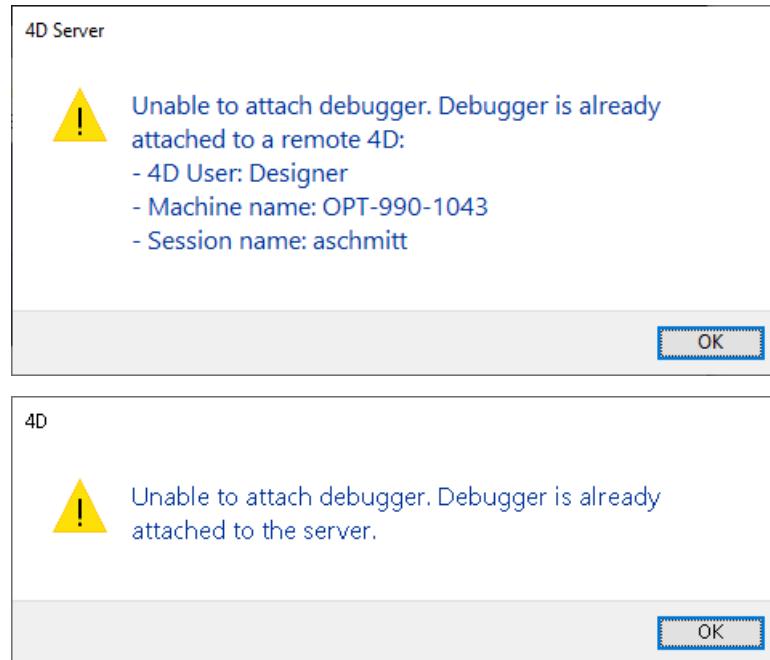
- On a remote 4D client, this option is named Attach Remote Debugger At Startup. When selected, the remote 4D client will automatically try to attach the remote debugger at each subsequent connection to the same 4D Server database. If the attachment is accepted (see [Rejected attachment requests](#)), the remote debugger is automatically attached to the remote 4D client and the Detach Remote Debugger option is displayed.

This setting is applied per project and is stored locally in the `.4DPreferences` file.

## Peticiones de adjuntos rechazadas

While the debugger is already attached to a remote 4D client or to 4D Server, no other machine can attach the debugger.

If a machine tries to attach the debugger while it is already attached, the attachment is rejected and a dialog box appears:



Adjuntar el depurador en este caso requiere que:

- the attached debugger is detached from the server or from the remote 4D client using respectively the Detach debugger or Detach remote debugger menu command,
- se cierra la sesión del cliente 4D remoto adjunto.

# Descripción de los archivos de historial

Las aplicaciones 4D pueden generar varios archivos de historial que son útiles para depurar u optimizar su ejecución. Los históricos suelen iniciarse o detenerse utilizando selectores de los comandos **SET DATABASE PARAMETER** o **WEB SET OPTION** y se almacenan en la carpeta [Logs folder](#) del proyecto.

La información histórica debe ser analizada para detectar y solucionar los problemas. Esta sección ofrece una descripción completa de los siguientes archivos de registro:

- [4DRequestsLog.txt](#)
- [4DRequestsLog\\_ProcessInfo.txt](#)
- [HTTPDebugLog.txt](#)
- [4DDebugLog.txt \(standard & tabular\)](#)
- [4DDiagnosticLog.txt](#)
- [4DIMAPLog.txt](#)
- [4DPOP3Log.txt](#)
- [4DSMTPLog.txt](#)
- [Archivo de historial de peticiones ORDA clientes](#)

Cuando un archivo de historial puede generarse tanto en 4D Server como en el cliente remoto, se añade la palabra "Server" al nombre del archivo de historial del lado del servidor, por ejemplo "4DRequestsLogServer.txt"

Los archivos de historial comparten algunos campos para que pueda establecer una cronología y hacer conexiones entre las entradas mientras depura:

- `sequence_number` : este número es único en todos los registros de depuración y se incrementa para cada nueva entrada cualquiera que sea el archivo de historial, para que pueda conocer la secuencia exacta de las operaciones.
- `connection_uuid` : para cada proceso 4D creado en un cliente 4D que se conecte a un servidor, este UUID de conexión se registra tanto del lado del servidor como del cliente. Permite identificar fácilmente el cliente remoto que lanzó cada proceso.

## 4DRequestsLog.txt

Este archivo de historial registra las solicitudes estándar llevadas a cabo por la máquina 4D Server o la máquina remota 4D que ejecutó el comando (excluyendo las solicitudes web).

Como iniciar este historial:

- en el servidor:

```
SET DATABASE PARAMETER(4D Server log recording;1)
//del lado del servidor
```

- en un cliente:

```
SET DATABASE PARAMETER(Client Log Recording;1)
///del lado remoto
```

Esta instrucción también inicia el archivo de historial [4DRequestsLog\\_ProcessInfo.txt](#).

## Encabezados

Este archivo comienza con los siguientes encabezados:

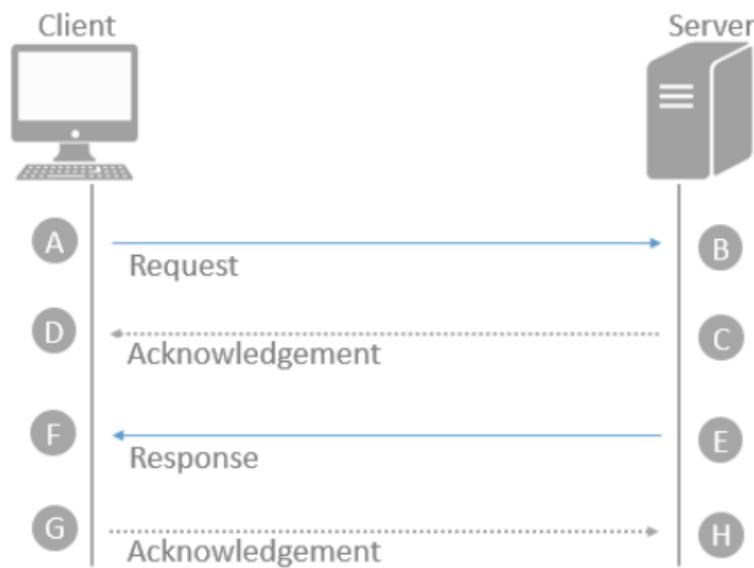
- Log Session Identifier (Identificador de sesión de historial)
- Nombre del servidor que aloja la aplicación
- Nombre de usuario: nombre de usuario en el sistema operativo que ejecutó la aplicación 4D en el servidor.

## Contenido

Para cada petición, se registran los siguientes campos:

Nombre del campo	Descripción
sequence_number	Número de operación único y secuencial en la sesión de historial
time	Fecha y hora utilizando el formato ISO 8601: 'YYYY-MM-DDTHH:MM:SS.mmm'
systemid	ID del sistema
component	Firma del componente (por ejemplo, "4SQLS" o "dbmg")
process_info_	corresponde al campo "index" en el archivo de historial 4DRequestsLog_ProcessInfo.txt, y permite vincular una petición a un proceso.
request	ID de petición en modo remoto cadena de mensajes para las peticiones SQL o mensajes LOG EVENT
bytes_in	Número de bytes recibidos
bytes_out	Número de bytes enviados
server_duration   exec_duration	Depende de dónde se genere el registro: <ul style="list-style-type: none"> <li>• <i>server_duration</i> cuando se genera en el cliente --Tiempo que tarda el servidor en procesar la petición y devolver una respuesta en microsegundos. B a F en la imagen de abajo, O</li> <li>• <i>exec_duration</i> cuando se genera en el servidor --Tiempo en microsegundos que tarda el servidor en procesar la petición. B a E en la imagen de abajo.</li> </ul>
write_duration	Tiempo de envío en microsegundos: <ul style="list-style-type: none"> <li>• La petición (cuando se ejecuta en el cliente). A a B en la imagen de abajo.</li> <li>• Respuesta (cuando se ejecuta en el servidor). E a F en la imagen de abajo.</li> </ul>
task_kind	Apropiativo o cooperativo (respectivamente "p" o "c")
rtt	Tiempo estimado en microsegundos para que el cliente envíe la solicitud y el servidor la acuse de recibo. De la A a la D y de la E a la H en la imagen de abajo. <ul style="list-style-type: none"> <li>• Sólo se mide cuando se utiliza la capa de red ServerNet, devuelve 0 cuando se utiliza con la capa de red heredada.</li> <li>• Para las versiones de Windows anteriores a Windows 10 o Windows Server 2016, la llamada devolverá 0.</li> </ul>

Flujo de solicitudes:



## 4DRequestsLog\_ProcessInfo.txt

Este archivo de historial registra la información de cada proceso creado en la máquina 4D Server o en la máquina remota 4D que ejecutó el comando (excluyendo las solicitudes web).

Como iniciar este historial:

- en el servidor:

```
SET DATABASE PARAMETER(4D Server log recording;1) //lado servidor
```

- en un cliente:

```
SET DATABASE PARAMETER(Client Log Recording;1) ////del lado remoto
```

Esta instrucción también inicia el archivo de historial [4DRequestsLog.txt](#).

### Encabezados

Este archivo comienza con los siguientes encabezados:

- Log Session Identifier (Identificador de sesión de historial)
- Nombre del servidor que aloja la aplicación
- Nombre de usuario: nombre de usuario en el sistema operativo que ejecutó la aplicación 4D en el servidor.

### Contenido

Para cada proceso, se registran los siguientes campos:

Nombre del campo	Descripción
sequence_number	Número de operación único y secuencial en la sesión de historial
time	Fecha y hora utilizando el formato ISO 8601: "YYYY-MM-DDTHH:MM:SS.mmm"
process_info_index	Número de proceso único y secuencial
CDB4DBaseContext	UUID del contexto de base del componente DB4D
systemid	ID del sistema
server_process_id	ID del proceso en el servidor
remote_process_id	ID del proceso en el cliente
process_name	Nombre del proceso
cID	Identificador de la conexión 4D
uID	Identificador del cliente 4D
IP Client	Dirección IPv4/IPv6
host_name	Nombre de host del cliente
user_name	Nombre de conexión usuario en el cliente
connection_uuid	Identificador UUID de proceso de conexión
server_process_unique_id	ID único del proceso en el servidor

## HTTPDebugLog.txt

Este archivo de historial registra cada petición HTTP y cada respuesta en modo crudo. Se registran las solicitudes completas, incluidos los encabezados; opcionalmente, también se pueden registrar las partes del cuerpo.

Como iniciar este historial:

```
WEB SET OPTION(Web debug log;wdl enable without body)
//otros valores están disponibles
```

Los siguientes campos se registran tanto para la solicitud como para la respuesta:

Nombre del campo	Descripción
SocketID	ID del socket utilizado para la comunicación
PeerIP	Dirección IPv4 del host (cliente)
PeerPort	Puerto utilizado por el host (cliente)
TimeStamp	Timestamp en milisegundos (desde el inicio del sistema)
ConnectionID	Conexión UUID (UUID del VTCPSocket utilizado para la comunicación)
SequenceNumber	Número de operación único y secuencial en la sesión de historial

## 4DDebugLog.txt (standard)

Este archivo de historial registra cada evento que se produce a nivel de programación 4D. El modo estándar ofrece una visión básica de los eventos.

Como iniciar este historial:

```

SET DATABASE PARAMETER(Debug Log Recording;2)
//estándar, todos los procesos

SET DATABASE PARAMETER(Current process debug log recording;2)
//estándar, sólo el proceso actual

```

Los siguientes campos se registran para cada evento:

Columna #	Descripción
1	Número de operación único y secuencial en la sesión de historial
2	Fecha y hora en el formato ISO 8601: (YYYY-MM-DDThh:mm:ss.mmm)
3	ID proceso (p=xx) e ID único proceso (puid=xx)
4	Nivel de stack
5	Puede ser Nombre del comando/Nombre del método/Mensaje/Información de inicio y parada de la tarea/Nombre, evento o callback plugin / UUID conexión
6	Tiempo de la operación de conexión en milisegundos

## 4DDebugLog.txt (tabular)

Este archivo de historial registra cada evento que se produce a nivel de programación 4D en un formato compacto y con pestañas que incluye información adicional (en comparación con el formato estándar).

Como iniciar este historial:

```

SET DATABASE PARAMETER(Debug Log Recording;2+4)
//formato tabular extendido, todos los procesos

SET DATABASE PARAMETER(Current process debug log recording;2+4)
//extendido, sólo el proceso actual

```

Los siguientes campos se registran para cada evento:

Columna #	Nombre del campo	Descripción
1	sequence_number	Número de operación único y secuencial en la sesión de historial
2	time	Fecha y hora en el formato ISO 8601: (YYYY-MM-DDThh:mm:ss.mmm)
3	ProcessID	ID del Proceso
4	unique_processID	ID único del proceso
5	stack_level	Nivel de stack
6	operation_type	<p>Tipo de operación histórico. Este valor puede ser un valor absoluto:</p> <ol style="list-style-type: none"> <li>1. Comando</li> <li>2. Método (método de proyecto, método base, etc.)</li> <li>3. Mensaje (enviado por el comando <a href="#">LOG EVENT</a> únicamente)</li> <li>4. PluginMessage</li> <li>5. PluginEvent</li> <li>6. PluginCommand</li> <li>7. PluginCallback</li> <li>8. Tarea</li> <li>9. Método miembro (método adjunto a una colección o a un objeto)</li> </ol> <p>Al cerrar un nivel de pila, las columnas <code>operation_type</code>, <code>operation</code> y <code>operation_parameters</code> tienen el mismo valor que el nivel de pila de apertura registrado en la columna <code>stack_opening_sequence_number</code>. Por ejemplo:</p> <ol style="list-style-type: none"> <li>1. 121 15:16:50:777 5 8 1 2 CallMethod Parameters 0</li> <li>2. 122 15:16:50:777 5 8 2 1 283 0</li> <li>3. 123 15:16:50:777 5 8 2 1 283 0 122 3</li> <li>4. 124 15:16:50:777 5 8 1 2 CallMethod Parameters 0 121 61</li> </ol> <p>La primera y la segunda línea abren el nivel de la pila, la tercera y la cuarta lo cierran. Los valores de las columnas 6, 7 y 8 se repiten en la línea del nivel de pila de cierre. La columna 10 contiene los números de secuencia de apertura del nivel de pila, es decir, 122 para la tercera línea y 121 para la cuarta.</p>
7	operation	Puede representar (según el tipo de operación): <ul style="list-style-type: none"> <li>• un ID de comando de lenguaje (cuando tipo=1)</li> <li>• un nombre de método (cuando tipo=2)</li> <li>• una combinación de pluginIndex;pluginCommand (cuando tipo=4, 5, 6 o 7). Puede contener algo como '3;2'</li> <li>• una UUID task connection (cuando tipo = 8)</li> </ul>
8	operation_parameters	Parámetros pasados a comandos, métodos o plugins
9	form_event	Evento formulario si lo hay; vacío en otros casos (supongamos que la columna se utiliza cuando se ejecuta el código en un método formulario o en un método objeto)
10	stack_opening_sequence_number	Sólo para los niveles de pila de cierre: número de secuencia del nivel de pila de apertura correspondiente
11	stack_level_execution_time	Sólo cuando se cierra el nivel de la pila: el tiempo transcurrido en microsegundos de la acción registrada actualmente (ver la décima columna en las líneas 123 y 124 del registro anterior)

## 4DDiagnosticLog.txt

Este archivo de historial registra muchos eventos relacionados con el funcionamiento interno de la aplicación y es legible para las personas. Puede incluir información personalizada en este archivo utilizando el comando [LOG EVENT](#).

Como iniciar este historial:

```
SET DATABASE PARAMETER(Diagnostic log recording;1) //iniciar registro
```

Los siguientes campos se registran para cada evento:

Nombre del campo	Descripción
sequenceNumber	Número de operación único y secuencial en la sesión de historial
timestamp	Fecha y hora en el formato ISO 8601: (YYYY-MM-DDThh:mm:ss.mmm)
loggerID	Opcional
componentSignature	Opcional - firma del componente interno
messageLevel	Información, avisos, errores
message	Descripción de la entrada del historial

Dependiendo del evento, se pueden incluir otros campos en el registro, como la tarea, socket, etc.

### Cómo activar el archivo

The *4DDiagnosticLog.txt* file can log different levels of messages, from `ERROR` (most important) to `TRACE` (less important). By default, the `INFO` level is set, which means that the file will log only important events, including errors and unexpected results (see below).

You can select the level of messages using the `Diagnostic log level` selector of the [SET DATABASE PARAMETER](#) command, depending on your needs. When you select a level, levels above (which are more important) are implicitly selected also. Los siguientes niveles están disponibles:

Columna #	Descripción	Cuando se selecciona, incluye
ERROR	Una parte de la aplicación no funciona	ERROR
WARN	Potential error, use of a deprecated function, poor uses, undesirable or unexpected situation	ERROR, WARN
INFO	ID Proceso 4D	ERROR, WARN, INFO
DEBUG	Detail of application flow (for 4D technical services)	ERROR, WARN, INFO, DEBUG
TRACE	Other internal information (for 4D technical services)	ERROR, WARN, INFO, DEBUG, TRACE

## 4DSMTPLog.txt, 4DPOP3Log.txt y 4DIMAPLog.txt

Estos archivos de registro registran cada intercambio entre la aplicación 4D y el servidor de correo (SMTP, POP3, IMAP) que ha sido iniciado por los siguientes comandos:

- SMTP - [SMTP New transporter](#)
- POP3 - [POP3 New transporter](#)
- IMAP - [IMAP New transporter](#)

Los archivos de historial pueden producirse en dos versiones:

- una versión normal:

- archivos llamados 4DSMTPLog.txt, 4DPOP3Log.txt, o 4DIMAPLog.txt
- sin adjuntos
- utiliza un reciclaje automático de archivos circulares cada 10 MB
- destinado a la depuración habitual

Para iniciar este historial:

```
SET DATABASE PARAMETER(SMTP Log;1) //iniciar log SMTP
SET DATABASE PARAMETER(POP3 Log;1) //iniciar log POP3
SET DATABASE PARAMETER(IMAP Log;1) //iniciar log IMAP
```

4D Server: clic en el botón Iniciar los historiales de peticiones y de depuración en la página [Mantenimiento](#) o de la ventana de administración de 4D Server.

Esta ruta al historial es devuelta por el comando `Get 4D file`.

- una versión extendida:
  - adjunto(s) incluido(s) no hay reciclaje automático
  - nombre personalizado
  - reservado para fines específicos

Para iniciar este historial:

```
$server:=New object
...
//SMTP
$server.logFile:="MySMTPAuthLog.txt"
$transporter:=SMTP New transporter($server)

// POP3
$server.logFile:="MyPOP3AuthLog.txt"
$transporter:=POP3 New transporter($server)

//IMAP
$server.logFile:="MyIMAPAuthLog.txt"
$transporter:=IMAP New transporter($server)
```

## Contenido

Para cada petición, se registran los siguientes campos:

Columna #	Descripción
1	Número de operación único y secuencial en la sesión de historial
2	Fecha y hora en el formato RFC3339 (yyyy-mm-ddThh:mm:ss.ms)
3	ID Proceso 4D
4	ID único del proceso
5	<ul style="list-style-type: none"> <li>• Información de inicio de sesión SMTP, POP3 o IMAP, incluyendo el nombre del servidor, el número de puerto TCP utilizado para conectarse al servidor SMTP, POP3 o IMAP y el estado de TLS, o</li> <li>• datos intercambiados entre el servidor y el cliente, empezando por "S &lt;" (datos recibidos del servidor SMTP, POP3 o IMAP) o "C &gt;" (datos enviados por el cliente SMTP, POP3 o IMAP): lista de modos de autenticación enviada por el servidor y modo de autenticación seleccionado, cualquier error notificado por el servidor SMTP, POP3 o IMAP, información del encabezado del correo enviado (sólo versión estándar) y si el correo se guarda en el servidor, o</li> <li>• Información de cierre de sesión SMTP, POP3 o IMAP.</li> </ul>

## Peticiones cliente ORDA

Este diario registra cada petición de ORDA enviada desde una máquina remota. Puede dirigir la información de registro a la memoria o a un archivo en el disco. El nombre y la ubicación de este archivo de registro son de su elección.

Como iniciar este historial:

```
//a ejecutar en una máquina remota
ds.startRequestLog(File("/PACKAGE/Logs/ordaLog.txt"))
//también puede enviarse a la memoria
```

Si desea utilizar el número de secuencia única en el registro de peticiones de ORDA, debe activarlo:

```
//a ejecutar en una máquina remota

SET DATABASE PARAMETER(Client Log Recording;1)
//para activar el número de secuencia del historial

ds.startRequestLog(File("/PACKAGE/Logs/ordaLog.txt"))
//también puede enviarse a la memoria

SET DATABASE PARAMETER(Client Log Recording;0)
//desactiva el número de secuencia
```

Los siguientes campos se registran para cada petición:

Nombre del campo	Descripción	Ejemplo
sequenceNumber	Número de operación único y secuencial en la sesión de historial	104
url	URL de la petición ORDA efectuada por el cliente	"rest/Persons(30001)"
startTime	Fecha y hora de inicio en formato ISO 8601	"2019-05-28T08:25:12.346Z"
endTime	Fecha y hora final en formato ISO 8601	"2019-05-28T08:25:12.371Z"
duration	Duración del procesamiento cliente (ms)	25
response	Objeto respuesta del servidor	{"status":200,"body": {"__entityModel":"Persons", [...]}}

# Utilización de un archivo de configuración de log

You can use a log configuration file to easily manage log recording in a production environment. This file is preconfigured by the developer. Typically, it can be sent to customers so that they just need to select it or copy it in a local folder. Once enabled, the log configuration file triggers the recording of specific logs.

## Cómo activar el archivo

There are several ways to enable the log configuration file:

- On 4D Server with interface, you can open the Maintenance page and click on the [Load logs configuration file](#) button, then select the file. In this case, you can use any name for the configuration file. Se activa inmediatamente en el servidor.
- You can copy the log configuration file in the [Settings folder](#) of the project. In this case, the file must be named `logConfig.json`. It is enabled at project startup (only on the server in client/server).
- With a built application, you can copy the `logConfig.json` file in the following folder:
  - Windows: `Users\[userName]\AppData\Roaming\[application]`
  - macOS: `/Users/[userName]/Library/ApplicationSupport/[application]`

If you want to enable the log configuration file for all projects in stand-alone, server and remote 4D applications, you can copy the `logConfig.json` file in the following folder: - Windows: `Users\[userName]\AppData\Roaming\4D` or `\4D Server` - macOS:

`/Users/[userName]/Library/ApplicationSupport/4D` or `/4D Server`

## Descripción del archivo JSON

The log configuration file is a `.json` file that can contain the following properties:

```
{  
    "$schema": "http://json-schema.org/draft-07/schema",  
    "title": "Logs Configuration File",  
    "description": "A file that controls the state of different types of logs in 4D clients and servers"  
    "type": "object",  
    "properties": {  
        "forceLoggingConfiguration": {  
            "description": "Forcing the logs configuration described in the file ingoring changes coming  
            "type": "boolean",  
            "default": true  
        },  
        "requestLogs": {  
            "description": "Configuration for request logs",  
            "type": "object",  
            "properties": {  
                "clientState": {  
                    "description": "Enable/Disable client request logs (from 0 to N)",  
                    "type": "integer",  
                    "minimum": 0  
                },  
                "serverState": {  
                    "description": "Enable/Disable server request logs (from 0 to N)",  
                    "type": "integer",  
                    "minimum": 0  
                }  
            }  
        },  
        "debugLogs": {  
            "description": "Configuration for debug logs",  
            "type": "object",  
            "properties": {  
                "commandList": {  
                    "description": "Commands to log or not log".  
                }  
            }  
        }  
    }  
}
```

```
        "type": "array",
        "items": {
            "type": "string"
        },
        "minItems": 1,
        "uniqueItems": true
    },
    "state": {
        "description": "integer to specify type of debuglog and options",
        "type": "integer",
        "minimum": 0
    }
}
},
"diagnosticLogs": {
    "description": "Configuration for debug logs",
    "type": "object",
    "properties": {
        "state": {
            "description": "Enable/Disable diagnostic logs 0 or 1 (0 = do not record, 1 = record",
            "type": "integer",
            "minimum": 0
        }
    }
},
"httpDebugLogs": {
    "description": "Configuration for http debug logs",
    "type": "object",
    "properties": {
        "level": {
            "description": "Configure http request logs",
            "type": "integer",
            "minimum": 0,
            "maximum": 7
        },
        "state": {
            "description": "Enable/Disable recording of web requests",
            "type": "integer",
            "minimum": 0,
            "maximum": 4
        }
    }
},
"POP3Logs": {
    "description": "Configuration for POP3 logs",
    "type": "object",
    "properties": {
        "state": {
            "description": "Enable/Disable POP3 logs (from 0 to N)",
            "type": "integer",
            "minimum": 0
        }
    }
},
"SMTPLogs": {
    "description": "Configuration for SMTP logs",
    "type": "object",
    "properties": {
        "state": {
            "description": "Enable/Disable SMTP log recording (form 0 to N)",
            "type": "integer",
            "minimum": 0
        }
    }
}
```

```
},
"IMAPLogs": {
    "description": "Configuration for IMAP logs",
    "type": "object",
    "properties": {
        "state": {
            "description": "Enable/Disable IMAP log recording (form 0 to N)",
            "type": "integer"
        }
    }
},
"ORDALogs": {
    "description": "Configuration for ORDA logs",
    "type": "object",
    "properties": {
        "state": {
            "description": "Enable/Disable ORDA logs (0 or 1)",
            "type": "integer"
        },
        "filename": {
            "type": "string"
        }
    }
}
}
```

## Ejemplo

Este es un ejemplo de archivo de configuración de log:

```
{  
    "forceLoggingConfiguration": false,  
    "requestLogs": {  
        "clientState": 1,  
        "serverState": 1  
    },  
    "debugLogs": {  
        "commandList": ["322", "311", "112"],  
        "state": 4  
    },  
    "diagnosticLogs": {  
        "state" : 1  
    },  
    "httpDebugLogs": {  
        "level": 5,  
        "state" : 1  
    },  
    "POP3Logs": {  
        "state" : 1  
    },  
    "SMTPLogs": {  
        "state" : 1  
    },  
    "IMAPLogs": {  
        "state" : 1  
    },  
    "ORDALogs": {  
        "state" : 1,  
        "filename": "ORDALog.txt"  
    }  
}
```

# Etiquetas de transformación

4D provides a set of transformation tags which allow you to insert references to 4D variables or expressions, or to perform different types of processing within a source text, referred to as a "template". These tags are interpreted when the source text is executed and generate an output text.

This principle is used in particular by the 4D Web server to build [web template pages](#).

These tags are generally to be inserted as HTML type comments (`<!--#Tag Contents-->`) but an [xml-compliant alternative syntax](#) is available for some of them.

Es posible mezclar varios tipos de etiquetas. For example, the following HTML structure is entirely feasible:

```
<HTML>
<BODY>
<!--#4DSCRIPT/PRE_PROCESS--> (Method call)
<!--#4DIF (myvar=1)--> (If condition)
    <!--#4DINCLUDE banner1.html--> (Subpage insertion)
<!--#4DENDIF--> (End if)
<!--#4DIF (mtvar=2)-->
    <!--#4DINCLUDE banner2.html-->
<!--#4DENDIF-->

<!--#4DLOOP [TABLE]--> (Loop on the current selection)
<!--#4DIF ([TABLE]ValNum>10)--> (If [TABLE]ValNum>10)
    <!--#4DINCLUDE subpage.html--> (Subpage insertion)
<!--#4DELSE--> (Else)
    <B>Value: <!--#4DTEXT [TABLE]ValNum--></B><BR> (Field display)
<!--#4DENDIF-->
<!--#4DENDLOOP--> ] (End for)
</BODY>
</HTML>
```

## Principios de uso de las etiquetas

### Parsing

Parsing the contents of a *template* source is done in two contexts:

- Using the `PROCESS 4D TAGS` command; this command accepts a *template* as input, as well as optional parameters and returns a text resulting from the processing.
- Using 4D's integrated HTTP server: [template pages](#) sent by means of the `WEB SEND FILE` (.htm, .html, .shtm, .shtml), `WEB SEND BLOB` (text/html type BLOB), `WEB SEND TEXT` commands, or called using URLs. In this last case, for reasons of optimization, pages that are suffixed with ".htm" and ".html" are NOT parsed. In order to parse HTML pages in this case, you must add the suffix ".shtm" or ".shtml" (for example, <http://www.server.com/dir/page.shtm>).

### Tratamiento recursivo

4D tags are interpreted recursively: 4D always attempts to reinterpret the result of a transformation and, if a new transformation has taken place, an additional interpretation is performed, and so on until the product obtained no longer requires any further transformation. Por ejemplo, dada la siguiente instrucción:

```
<!--#4DHTML [Mail]Letter_type-->
```

If the `[Mail]Letter_type` text field itself contains a tag, for example `<!--#4DSCRIPT/m_Gender-->`, this tag will be evaluated recursively after the interpretation of the 4DHTML tag.

This powerful principle meets most needs related to text transformation. Note, however, that in some cases this can also allow malicious code to be inserted in the web context, [which can be avoided](#).

## Identificadores con tokens

To ensure the correct evaluation of expressions processed via tags, regardless of the language or 4D version, it's recommended to use the tokenized syntax for elements whose name may vary over versions (commands, tables, fields, constants). For example, to insert the `Current time` command, enter `Current time: C178`.

## Utilizando el "." como separador decimal

4D always uses the period character (.) as a decimal separator when evaluating a numerical expression using a 4D tag `4DTEXT`, `4DHTML`, and `4DEVAL`. Los parámetros regionales se ignoran. This feature facilitates code maintenance and compatibility between 4D languages and versions.

## 4DBASE

Syntax: `<!--#4DBASE folderPath-->`

The `<!--#4DBASE -->` tag designates the working directory to be used by the `<!--#4DINCLUDE-->` tag.

When it is called in a Web page, the `<!--#4DBASE -->` tag modifies all subsequent `<!--#4DINCLUDE-->` calls on this page, until the next `<!--#4DBASE -->`, if any. If the ` folder is modified from within an included file, it retrieves its original value from the parent file.

The `folderPath` parameter must contain a pathname relative to the current page and it must end with a slash (/). The designated folder must be located inside the Web folder.

Pass the "WEBFOLDER" keyword to restore the default path (relative to the page).

The following code, which must specify a relative path for each call:

```
<!--#4DINCLUDE subpage.html-->
<!--#4DINCLUDE folder/subpage1.html-->
<!--#4DINCLUDE folder/subpage2.html-->
<!--#4DINCLUDE folder/subpage3.html-->
<!--#4DINCLUDE ../folder/subpage.html-->
```

... es equivalente a:

```
<!--#4DINCLUDE subpage.html-->
<!--#4DBASE folder-->
<!--#4DINCLUDE subpage1.html-->
<!--#4DINCLUDE subpage2.html-->
<!--#4DINCLUDE subpage3.html-->
<!--#4DBASE ../folder-->
<!--#4DINCLUDE subpage.html-->
<!--#4DBASE WEBFOLDER-->
```

Por ejemplo, para definir un directorio para la página de inicio:

```
/* Index.html */
<!--#4DIF LangFR=True-->
    <!--#4DBASE FR/-->
<!--#4DELSE-->
    <!--#4DBASE US/-->
<!--#4DENDIF-->
<!--#4DINCLUDE head.html-->
<!--#4DINCLUDE body.html-->
<!--#4DINCLUDE footer.html-->
```

In the "head.html" file, the current folder is modified through `<!--#4DBASE -->`, without this changing its value in "Index.html":

```
/* Head.htm */
/* the working directory here is relative to the included file (FR/ or US/) */
<!--#4DBASE Styles/-->
<!--#4DINCLUDE main.css-->
<!--#4DINCLUDE product.css-->
<!--#4DBASE Scripts/-->
<!--#4DINCLUDE main.js-->
<!--#4DINCLUDE product.js-->
```

## 4DCODE

Syntax: `<!--#4DCODE codeLines-->`

The `4DCODE` tag allows you to insert a multi-line 4D code block in a template.

When a `<!--#4DCODE` sequence is detected that is followed by a space, a CR or a LF character, 4D interprets all the lines of code up to the next `-->` sequence. The code block itself can contain carriage returns, line feeds, or both; it will be interpreted sequentially by 4D.

Por ejemplo, puede escribir en una plantilla:

```
<!--#4DCODE
//PARAMETERS initialization
C_OBJECT:C1216($graphParameters)
OB SET: C1220($graphParameters;"graphType";1)
$graphType:=1
//...your code here
If(OB Is defined: C1231($graphParameters;"graphType"))
    $graphType:=OB GET: C1224($graphParameters;"graphType")
    If($graphType=7)
        $nbSeries:=1
        If($nbValues>8)
            DELETE FROM ARRAY: C228 ($yValuesArrPtr{1}->;9;100000)
            $nbValues:=8
        End if
    End if
End if
-->
```

Aquí están las características de la etiqueta 4DCODE:

- The `TRACE` command is supported and activates the 4D debugger, thus allowing you to debug your template code.
- Any error will display the standard error dialog that lets the user stop code execution or enter debugging mode.
- The text in between `<!--#4DCODE` and `-->` is split into lines accepting any line-ending convention (cr, lf, or crlf).
- The text is tokenized within the context of the database that called `PROCESS 4D TAGS`. This is important for

recognition of project methods for example. The [Available through tags and 4D URLs \(4DACTION ...\)](#) method property is not taken into account.

- Even if the text always uses English-US, it is recommended to use the token syntax (:Cxxx) for command and constant names to protect against potential problems due to commands or constants being renamed from one version of 4D to another.

The fact that 4DCODE tags can call any of the 4D language commands or project methods could be seen as a security issue, especially when the database is available through HTTP. However, since it executes server-side code called from your own template files, the tag itself does not represent a security issue. In this context, as for any Web server, security is mainly handled at the level of remote accesses to server files.

## 4DEACH y 4DENDEACH

Syntax: `<!--#4DEACH variable in expression--> <!--#4DENDEACH-->`

The `<!--#4DEACH-->` comment allows iterating a specified item over all values of the *expression*. The item is set to a *variable* whose type depends on the *expression* type.

The `<!--#4DEACH-->` comment can iterate through three expression types:

- [collections](#): loop through each element of the collection,
- [entity selections](#): loop through each entity,
- [objects](#): loop through each object property.

The number of iterations is evaluated at startup and will not change during the processing. La adición o eliminación de elementos durante el bucle no suele ser recomendable, ya que puede resultar en redundancia o perdidas de iteraciones.

`<!--#4DEACH item in collection-->`

This syntax iterates on each *item* of the *collection*. The code portion located between `<!--#4DEACH -->` and `<!--#4DENDEACH-->` is repeated for each collection element.

The *item* parameter is a variable of the same type as the collection elements.

The collection must contain only elements of the same type, otherwise an error is returned as soon as the *item* variable is assigned the first mismatched value type.

The number of loops is based on the number of elements of the collection. At each iteration, the *item* variable is automatically filled with the matching element of the collection. Hay que tener en cuenta los siguientes puntos:

- If the *item* variable is of the object type or collection type (i.e. if *expression* is a collection of objects or of collections), modifying this variable will automatically modify the matching element of the collection (because objects and collections share the same references). Si la variable es de tipo escalar, sólo se modificará la variable.
- The *item* variable gets the same type as the first collection element. If any collection element is not of the same type as the variable, an error is generated and the loop stops.
- If the collection contains elements with a Null value, an error is generated if the *item* variable type does not support Null values (such as longint variables).

Ejemplo con una colección de valores escalares

`getNames` returns a collection of strings. The method has been declared as "[available through 4D tags and URLs](#)".

```


| Name           |
|----------------|
| #4DTEXT \$name |


```

## Ejemplo con una colección de objetos

`getSalesPersons` returns a collection of objects.

```


|                          |                                 |                                |
|--------------------------|---------------------------------|--------------------------------|
| #4DTEXT \$salesPerson.ID | #4DTEXT \$salesPerson.firstname | #4DTEXT \$salesPerson.lastname |
|--------------------------|---------------------------------|--------------------------------|


```

```
<!--#4DEACH entity in entitySelection-->
```

This syntax iterates on each `entity` of the `entitySelection`. The code portion located between `<!--#4DEACH -->` and `<!--#4DENDEACH-->` is repeated for each entity of the entity selection.

The `entity` parameter is an object variable of the entity selection class.

The number of loops is based on the number of entities of the entity selection. At each iteration, the `entity` object variable is automatically filled with the matching entity of the entity selection.

## Ejemplo con una tabla html

```


| ID | Name | Total purchase |
|----|------|----------------|
|----|------|----------------|


```

## Ejemplo con PROCESS 4D TAGS

```

var customers : cs.CustomersSelection
var $input; $output : Text

customers:=ds.Customers.all()
$input:="!--#4DEACH $cust in customers-->"
$input:=$input+"<!--#4DTEXT $cust.name -->"+Char(Carriage return)
$input:=$input+"<!--#4DENDEACH-->"
PROCESS 4D TAGS($input; $output)
TEXT TO DOCUMENT("customers.txt"; $output)

```

### <!--#4DEACH property in object-->

This syntax iterates on each *property* of the *object*. The code portion located between `<!--#4DEACH -->` and `<!--#4DENDEACH-->` is repeated for each property of the object.

The *property* parameter is a text variable automatically filled with the name of the currently processed property.

The properties of the object are processed according to their creation order. Durante el bucle, se pueden añadir o eliminar propiedades en el objeto, sin modificar el número de bucles que quedarán en función del número original de propiedades del objeto.

Ejemplo con las propiedades de un objeto

`getGamers` is a project method that returns an object like ("Mary"; 10; "Ann"; 20; "John"; 40) to figure gamer scores.

```





```

## 4DEVAL

Syntax: `<!--#4DEVAL expression-->`

Alternative syntax: `$4DEVAL(expression)`

The `4DEVAL` tag allows you to assess a 4D variable or expression. Like the `4DHTML` tag, `4DEVAL` does not escape HTML characters when returning text. However, unlike `4DHTML` or `4DTEXT`, `4DEVAL` allows you to execute any valid 4D statement, including assignments and expressions that do not return any value.

Por ejemplo, puede ejecutar:

```

$input:="!--#4DEVAL a:=42-->" //assignment
$input:=$input+"<!--#4DEVAL a+1-->" //calculation
PROCESS 4D TAGS($input;$output)
//$/output = "43"

```

In case of an error during interpretation, the text inserted will be in the form: `<!--#4DEVAL expr-->: ## error # error code .`

For security reasons, it is recommended to use the `4DTEXT` tag when processing data introduced from outside the application, in order to prevent the [insertion of malicious code](#).

## 4DHTML

Syntax: `<!--#4DHTML expression-->`

Alternative syntax: `$4DHTML(expression)`

The value of the 4D variable `vtSiteName` will be inserted in the HTML page when it is sent. This value is inserted as simple text, special HTML characters such as ">" are automatically escaped.

For example, here are the processing results of the 4D text variable myvar with the available tags:

Valor myvar	Etiquetas	Resultado
<code>myvar:="&lt;B&gt;"</code>	<code>&lt;!--#4DTEXT myvar--&gt;</code>	<code>&amp;lt;B&amp;gt;</code>
<code>myvar:="&lt;B&gt;"</code>	<code>&lt;!--#4DHTML myvar--&gt;</code>	<code>&lt;B&gt;</code>

In case of an interpretation error, the inserted text will be `<!--#4DHTML myvar--> : ## error # error code .`

For security reasons, it is recommended to use the `4DTEXT` tag when processing data introduced from outside the application, in order to prevent the [insertion of malicious code](#).

## 4DIF, 4DELSE, 4DELSEIF y 4DENDIF

Syntax: `<!--#4DIF expression--> { <!--#4DELSEIF expression2-->...<!--#4DELSEIF expressionN--> } <!--#4DELSE--> } <!--#4DENDIF-->`

Used with the `<!--#4DELSEIF-->` (optional), `<!--#4DELSE-->` (optional) and `<!--#4DENDIF-->` comments, the `<!--#4DIF expression-->` comment offers the possibility to execute portions of code conditionally.

The `expression` parameter can contain any valid 4D expression returning a Boolean value. It must be indicated within parenthesis and comply with the 4D syntax rules.

In case of an interpretation error, the text " `<!--#4DIF expression--> : A Boolean expression was expected`" is inserted instead of the contents located between `<!--#4DIF -->` and `<!--#4DENDIF-->`. Likewise, if there are not as many `<!--#4DENDIF-->` as `<!--#4DIF -->`, the text " `<!--#4DIF expression--> : 4DENDIF expected`" is inserted instead of the contents located between `<!--#4DIF -->` and `<!--#4DENDIF-->`.

In case of an interpretation error, the text " `<!--#4DIF expression--> : A Boolean expression was expected`" is inserted instead of the contents located between `<!--#4DIF -->` and `<!--#4DENDIF-->`. The `<!--#4DIF expression--> ... <!--#4DENDIF-->` blocks can be nested in several levels. Like in 4D, each `<!--#4DIF expression-->` must match a `<!--#4DENDIF-->`.

Using the `<!--#4DELSEIF-->` tag, you can test an unlimited number of conditions. Only the code that follows the first condition evaluated as `True` is executed. If no conditions are true, no statement is executed (if there is no final `<!--#4DELSE-->`). You can use a tag after the last . If all the conditions are false, the statements following the are executed.

Los dos códigos siguientes son equivalentes.

Código utilizando sólo 4DELSE:

```

<!--#4DIF Condition1-->
/* Condition1 is true*/
<!--#4ELSE-->
<!--#4DIF Condition2-->
/* Condition2 is true*/
<!--#4ELSE-->
<!--#4DIF Condition3-->
/* Condition3 is true */
<!--#4ELSE-->
/*None of the conditions are true*/
<!--#4ENDIF-->
<!--#4ENDIF-->
<!--#4ENDIF-->

```

Código similar utilizando la etiqueta `4ELSEIF` :

```

<!--#4DIF Condition1-->
/* Condition1 is true*/
<!--#4ELSEIF Condition2-->
/* Condition2 is true*/
<!--#4ELSEIF Condition3-->
/* Condition3 is true */
<!--#4ELSE-->
/* None of the conditions are true*/
<!--#4ENDIF-->

```

This example of code inserted in a static HTML page displays a different label according the `vname#""` expression result:

```

<BODY>
...
<!--#4DIF (vname#"")-->
Names starting with <!--#4DTEXT vname-->.
<!--#4ELSE-->
No name has been found.
<!--#4ENDIF-->
...
</BODY>

```

This example inserts different pages depending on which user is connected:

```

<!--#4DIF LoggedIn=False-->
    <!--#4DINCLUDE Login.htm -->
<!--#4ELSEIF User="Admin" -->
    <!--#4DINCLUDE AdminPanel.htm -->
<!--#4ELSEIF User="Manager" -->
    <!--#4DINCLUDE SalesDashboard.htm -->
<!--#4ELSE-->
    <!--#4DINCLUDE ItemList.htm -->
<!--#4ENDIF-->

```

## 4DINCLUDE

Sintaxis: `<!--#4DINCLUDE path-->`

This tag is mainly designed to include an HTML page (indicated by the `path` parameter) in another HTML page. By default, only the body of the specified HTML page, i.e. the contents found within the `<body>` and `</body>` tags, is

included (the tags themselves are not included). This lets you avoid conflicts related to meta tags present in the headers.

However, if the HTML page specified does not contain `<body>` `</body>` tags, the entire page is included. It is up to you to verify the consistency of the meta tags.

The `<!--#4DINCLUDE -->` comment is very useful for tests (`<!--#4DIF-->`) or loops (`<!--#4DL00P-->`). It is very convenient to include banners according to a criteria or randomly. When including, regardless of the file name extension, 4D analyzes the called page and then inserts the contents (modified or not) in the page originating the `4DINCLUDE` call.

An included page with the `<!--#4DINCLUDE -->` comment is loaded in the Web server cache the same way as pages called via a URL or sent with the `WEB SEND FILE` command.

In `path`, put the path leading to the document to include. Warning: In the case of a `4DINCLUDE` call, the path is relative to the document being analyzed, that is, the "parent" document. Use the slash character (/) as a folder separator and the two dots (..) to go up one level (HTML syntax). When you use the `4DINCLUDE` tag with the `PROCESS 4D TAGS` command, the default folder is the project folder.

You can modify the default folder used by the `4DINCLUDE` tag in the current page, using the `<!--#4DBASE -->` tag (see below).

The number of `<!--#4DINCLUDE path-->` within a page is unlimited. However, the `<!--#4DINCLUDE path-->` calls can be made only at one level. This means that, for example, you cannot insert `<!--#4DINCLUDE mydoc3.html-->` in the `mydoc2.html` body page, which is called by `<!--#4DINCLUDE mydoc2-->` inserted in `mydoc1.html`. Furthermore, 4D verifies that inclusions are not recursive.

In case of error, the inserted text is "`<!--#4DINCLUDE path-->` :The document cannot be opened".

Ejemplos:

```
<!--#4DINCLUDE subpage.html-->
<!--#4DINCLUDE folder/subpage.html-->
<!--#4DINCLUDE ../folder/subpage.html-->
```

## 4DLOOP y 4DENDLOOP

Syntax: `<!--#4DL00P condition--> <!--#4DENDL00P-->`

This comment allows repetition of a portion of code as long as the condition is fulfilled. The portion is delimited by `<!--#4DL00P-->` and `<!--#4DENDL00P-->`.

The `<!--#4DL00P condition--> ... <!--#4DENDL00P-->` blocks can be nested. Like in 4D, each `<!--#4DL00P condition-->` must match a `<!--#4DENDL00P-->`.

Hay cinco tipos de condiciones:

```
<!--#4DL00P [table]-->
```

This syntax makes a loop for each record from the table current selection in the current process. The code portion located between the two comments is repeated for each current selection record.

When the `4DL00P` tag is used with a table, records are loaded in "Read only" mode.

El código siguiente:

```
<!--#4DLOOP [People]-->
<!--#4DTEXT [People]Name--> <!--#4DTEXT [People]Surname--><BR>
<!--#4DENDLOOP-->
```

... could be expressed in 4D language in the following way:

```
FIRST RECORD( [People])
While(Not(End selection([People])))
  ...
  NEXT RECORD( [People])
End while
```

```
<!--#4DLOOP array-->
```

Esta sintaxis hace un bucle para cada elemento del array. The array current item is increased when each code portion is repeated.

This syntax cannot be used with two dimension arrays. In this case, it is better to combine a method with nested loops.

The following code example:

```
<!--#4DLOOP arr_names-->
<!--#4DTEXT arr_names{arr_names}--><BR>
<!--#4DENDLOOP-->
```

... could be expressed in 4D language in the following way:

```
For($Elem;1;Size of array(arr_names))
  arr_names:=$Elem
  ...
End for
```

```
<!--#4DLOOP method-->
```

This syntax makes a loop as long as the method returns `True`. El método toma un tipo de parámetro Long Integer. First it is called with the value 0 to allow an initialization stage (if necessary); it is then called with the values 1 ,then 2, then 3 and so on, as long as it returns `True`.

For security reasons, within a Web process, the `On Web Authentication` database method can be called once just before the initialization stage (method execution with 0 as parameter). If the authentication is OK, the initialization stage will proceed.

`C_BOOLEAN($0)` y `C_LONGINT($1)` DEBE declararse dentro del método a efectos de compilación.

The following code example:

```
<!--#4DLOOP my_method-->
<!--#4DTEXT var--> <BR>
<!--#4DENDLOOP-->
```

... could be expressed in 4D language in the following way:

```

If(AuthenticationWebOK)
  If(my_method(0))
    $counter:=1
    While(my_method($counter))
      ...
      $counter:=$counter+1
    End while
  End if
End if

```

The `my_method` method can be as follows:

```

C_LONGINT($1)
C_BOOLEAN($0)
If($1=0) `Initialisation
  $0:=True
Else
  If($1<50)
    ...
    var:...
    $0:=True
  Else
    $0:=False `Detiene el bucle
  End if
End if

```

`<!--#4DL00P expression-->`

With this syntax, the `4DL00P` tag makes a loop as long as the *expression* returns `True`. The expression can be any valid Boolean expression and must contain a variable part to be evaluated in each loop to avoid infinite loops.

Por ejemplo, el siguiente código:

```

<!--#4DEVAL $i:=0-->
<!--#4DL00P ($i<4)-->
<!--#4DEVAL $i-->
<!--#4DEVAL $i:=$i+1-->
<!--#4DENDLOOP-->

```

...produce el siguiente resultado:

```

0
1
2
3

```

`<!--#4DL00P pointerArray-->`

In this case, the `4DL00P` tag works like it does with an array: it makes a loop for each element of the array referenced by the pointer. The current array element is increased each time the portion of code is repeated.

This syntax is useful when you pass an array pointer as a parameter to the `PROCESS 4D TAGS` command.

Ejemplo:

```

ARRAY TEXT($array;2)
$array{1}:="hello"
$array{2}:="world"
$input:="!--#4DEVAL $1-->"
$input:=$input+"!--#4DLOOP $2-->"
$input:=$input+"!--#4DEVAL $2->{$2->}--> "
$input:=$input+"!--#4DENDLOOP-->"
PROCESS 4D TAGS($input;$output;"elements = ";->$array)
// $output = "elements = hello world "

```

In case of an interpretation error, the text " <!--#4DLOOP expression--> : description" is inserted instead of the contents located between <!--#4DLOOP --> and <!--#4DENDLOOP--> .

Se pueden mostrar los siguientes mensajes:

- Tipo de expresión inesperado (error estándar);
- Nombre de tabla incorrecto (error en el nombre de la tabla);
- An array was expected (the variable is not an array or is a two dimension array);
- El método no existe;
- Error de sintaxis (cuando el método se está ejecutando);
- Access error (you do not have the appropriate access privileges to access the table or the method).
- 4DENDLOOP expected (the <!--#4DENDLOOP--> number does not match the <!--#4DLOOP --> ).

## 4SCRIPT/

Syntax: <!--#4SCRIPT/MethodName/MyParam-->

The **4SCRIPT** tag allows you to execute 4D methods when processing the template. The presence of the <!--#4SCRIPT/MethodName/MyParam--> tag as an HTML comment launches the execution of the **MethodName** method with the **Param** parameter as a string in **\$1**.

If the tag is called in the context of a Web process, when the page is loaded, 4D calls the **On Web Authentication** database method (if it exists). Si devuelve True, 4D ejecuta el método.

El método debe devolver el texto en **\$0**. If the string starts with the code character 1, it is considered as HTML (the same principle is true for the **4DHTML** tag).

For example, let's say that you insert the following comment "Today is <!--#4SCRIPT/MYMETH/MYPARAM-->" into a template Web page. When loading the page, 4D calls the **On Web Authentication** database method, then calls the **MYMETH** method and passes the string "/MYPARAM" as the parameter **\$1**. The method returns text in **\$0** (for example "12/31/21"); the expression " Today is<!--#4SCRIPT/MYMETH/MYPARAM--> " therefore becomes "Today is 12/31/21".

El método **MYMETH** es el siguiente:

```

//MYMETH
C_TEXT($0;$1) //Estos parámetros deben declararse siempre
$0:=String(Current date)

```

A method called by **4SCRIPT** must not call interface elements ( **DIALOG** , **ALERT** , etc.).

As 4D executes methods in their order of appearance, it is absolutely possible to call a method that sets the value of many variables that are referenced further in the document, whichever mode you are using. You can insert as many <!--#4SCRIPT...--> comments as you want in a template.

## 4DTEXT

Syntax: <!--#4DTEXT expression-->

Alternative syntax: \$4DTEXT(expression)

The tag <!--#4DTEXT expression--> allows you to insert a reference to a 4D variable or expression returning a value. Por ejemplo, si se escribe (en una página HTML):

```
<P>Welcome to <!--#4DTEXT vtSiteName-->!</P>
```

Just like the 4DTEXT tag, this tag lets you assess a 4D variable or expression that returns a value, and insert it as an HTML expression. This value is inserted as simple text, special HTML characters such as ">" are automatically escaped.

También puede insertar expresiones 4D. You can for example directly insert the contents of a field ( <!--#4DTEXT [tableName]fieldName--> ), an array element ( <!--#4DTEXT tabarr{1}--> ) or a method returning a value ( <!--#4DTEXT mymethod--> ). The expression conversion follows the same rules as the variable ones. Moreover, the expression must comply with 4D syntax rules.

For security reasons, it is recommended to use this tag when processing data introduced from outside the application, in order to prevent the [insertion of malicious code](#).

In case of an evaluation error, the inserted text will appear as <!--#4DTEXT myvar--> : ## error # error code .

- Debe utilizar las variables proceso.
- Puede mostrar el contenido de un campo imagen. However, it is not possible to display the content of a picture array item.
- It is possible to display the contents of an object field by means of a 4D formula. For example, you can write <!--#4DTEXT 0B Get: C1224([Rect]Desc;\\"color\\")--> .
- Normalmente se trabaja con variables de tipo texto. Sin embargo, también se pueden utilizar las variables BLOB. You just need to generate BLOBs in Text without length mode.

## Sintaxis alternativa para 4DTEXT, 4DHTML, 4DEVAL

Several existing 4D transformation tags can be expressed using a \$-based syntax:

\$4dtag (expression)

puede utilizarse en lugar de

```
<!--#4dtag expression-->
```

This alternative syntax is available only for tags used to return processed values:

- [4DTEXT](#)
- [4DHTML](#)
- [4DEVAL](#)

(Other tags, such as 4DIF or 4DSCRIPT, must be written with the regular syntax).

Por ejemplo, puede escribir:

```
$4DEVAL(UserName)
```

en lugar de:

```
<!--#4DEVAL(UserName)-->
```

The main advantage of this syntax is that it allows you to write XML-compliant templates. Some 4D developers need to

create and validate XML-based templates using standard XML parser tools. Since the "<" character is invalid in an XML attribute value, it was not possible to use the "<!-- -->" syntax of 4D tags without breaking the document syntax. On the other hand, escaping the "<" character will prevent 4D from interpreting the tags correctly.

For example, the following code would cause an XML parsing error because of the first "<" character in the attribute value:

```
<line x1="" y1=""/>
```

Utilizando la sintaxis \$, el siguiente código es validado por el analizador:

```
<line x1="$4DEVAL($x)" y1="$4DEVAL($graphY1)"/>
```

Note that \$4dtag and <--#4dtag --> are not strictly equivalent: unlike <--#4dtag -->, \$4dtag processing does not interpret 4D tags [recursively](#). Las etiquetas \$ siempre se evalúan una vez y el resultado se considera texto plano.

The reason for this difference is to prevent malicious code injection. As [explained below](#), it is strongly recommended to use 4DTEXT tags instead of 4DHMTL tags when handling user text to protect against unwanted reinterpretation of tags: with 4DTEXT, special characters such as "<" are escaped, thus any 4D tags using the <!--#4dtag expression --> syntax will lose their particular meaning. However, since 4DTEXT does not escape the \$ symbol, we decided to break support for recursion in order to prevent malicious injection using the \$4dtag (expression) syntax.

The following examples show the result of processing depending on the syntax and tag used:

```
// example 1
myName:="" //malicious injection
input:="My name is: <!--#4DHTML myName-->"
PROCESS 4D TAGS(input;output)
//4D will quit!
```

```
// example 2
myName:="" //malicious injection
input:="My name is: <!--#4DTEXT myName-->"
PROCESS 4D TAGS(input;output)
//output is "My name is: <!--#4DHTML QUIT 4D-->"
```

```
// example 3
myName:="$4DEVAL(QUIT 4D)" //malicious injection
input:="My name is: <!--#4DTEXT myName-->"
PROCESS 4D TAGS(input;output)
//output is "My name is: $4DEVAL(QUIT 4D)"
```

Note that the \$4dtag syntax supports matching pairs of enclosed quotes or parenthesis. For example, suppose that you need to evaluate the following complex (unrealistic) string:

```
String(1) + "\"(hello)\""
```

Puede escribir:

```
input:="$4DEVAL( String(1)+\"\\\"\"(hello)\\\"\\\"\")"
PROCESS 4D TAGS(input;output)
-->output is 1"(hello)"
```

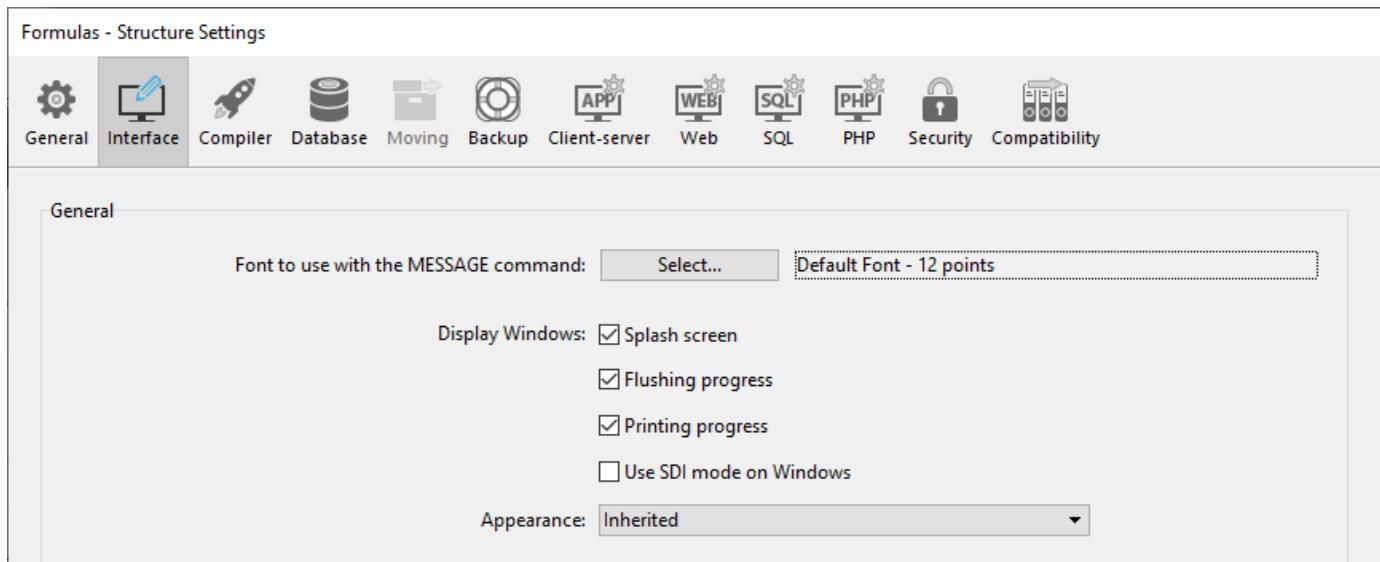


# Página interfaz

You use the Interface page to set various options related to the project interface.

## General

This area lets you set various options concerning display.



## Fuente a utilizar con el comando MESSAGE

Click Select... to set the font and size for the characters used by the `MESSAGE` command.

The default font and its size depend on the platform where 4D is running.

This property also affects the following parts of 4D:

- ciertas áreas de vista previa del Explorador
  - la regla del editor de formularios
- Other options configure the display of various windows in the Application mode.

- Splash screen: When this option is deselected, the [splash screen of the current menu bar](#) does not appear in the Application mode. When you hide this window, it is up to you to manage the display of all your windows by programming, for example in the `On Startup` database method.
- Flushing progress: When this option is checked, 4D displays a window at the bottom left of the screen while the data in the cache is flushed. Since this operation momentarily blocks user actions, displaying this window lets them know that flushing is underway.

You can set the [frequency for cache flushing](#) in Settings > Database > Memory.

- Printing progress: Lets you enable or disable the display of the printing progress dialog box when printing.
- Use SDI mode on Windows: When this option checked, 4D enables automatically the SDI mode (Single-Document Interface) in your merged application if executed in a supported context.

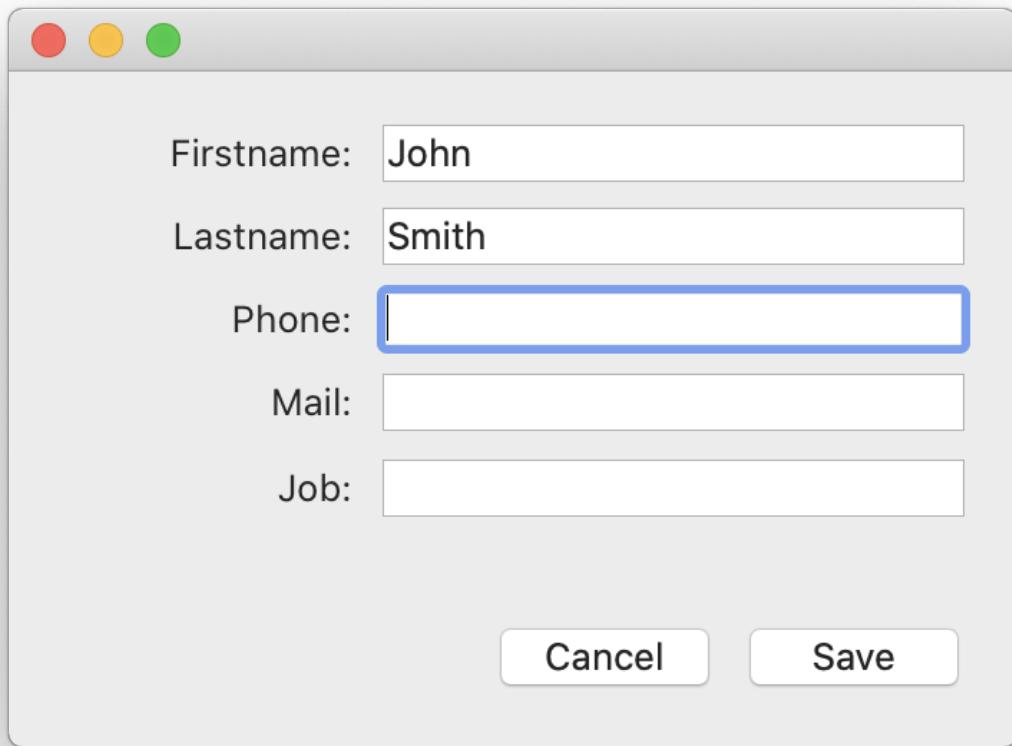
This option can be selected on macOS but will be ignored when the application is executed on this platform.

This menu lets you select the color scheme to use at the main application level. A color scheme defines a global set of interface colors for texts, backgrounds, windows, etc., used in your forms.

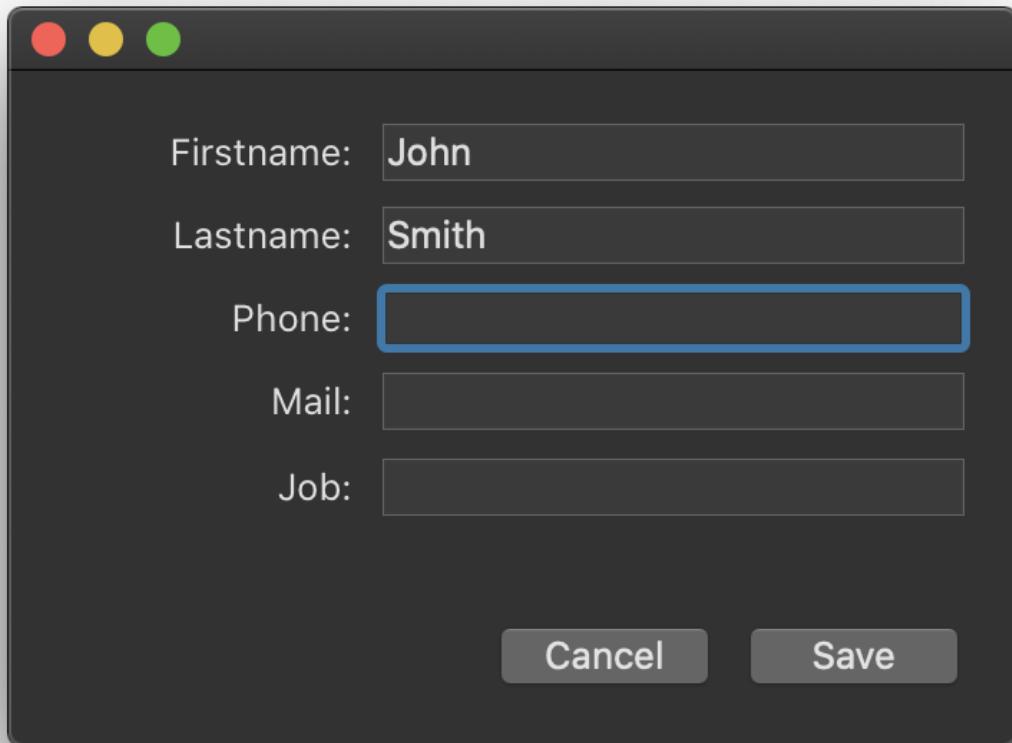
Esta opción sólo funciona en macOS. En Windows, se utiliza siempre el esquema "Light".

Los siguientes esquemas están disponibles:

- Light: the application will use the Default Light Theme



- Dark: the application will use the Default Dark Theme



- Inherited (default): the application will inherit from the higher priority level (i.e., OS user preferences)

Los temas por defecto pueden ser manejados utilizando CSS. For more information, please refer to the [Media Queries](#) section.

The main application scheme will be applied to forms by default. Sin embargo, se puede remplazar:

- by the [SET APPLICATION COLOR SCHEME](#) command at the working session level;
- using by the [Color Scheme](#) form property at each form level (highest priority level). Note: When printed, forms always use the "Light" scheme.

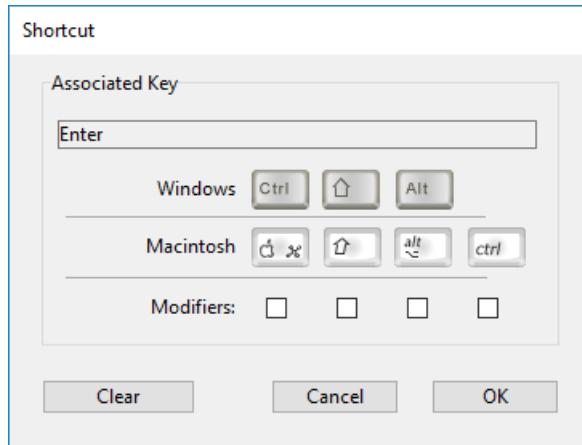
## Atajos

You use the Shortcuts area for viewing and modifying default shortcuts for three basic 4D form operations in your desktop applications. Estos atajos son idénticos para ambas plataformas. Key icons indicate the corresponding Windows and macOS keys.

Los accesos directos por defecto son los siguientes:

- Aceptación de formulario de entrada: Entrada
- Cancel input form: Esc
- Add to subform: Ctrl+Shift+ / (Windows) or Command+Shift+ / (macOS)

To change the shortcut of an operation, click the corresponding Edit button. Aparece la siguiente caja de diálogo:



To change the shortcut, type the new key combination on your keyboard and click OK. If you prefer not to have a shortcut for an operation, click Clear.

# Página Compilador

These parameters are detailed in the [Compiler Settings](#) section.

# Página Base de datos

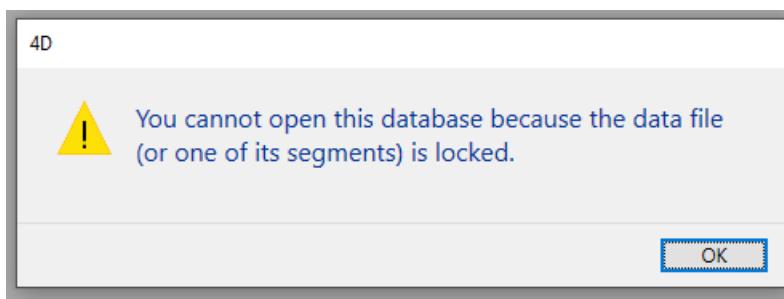
## Página Almacenamiento de datos

You use this page to configure data storage on disk for the 4D database.

### Parámetros generales

#### Autorizar el uso de archivos de datos de sólo lectura

This option allows configuration of the application operation when opening a locked data file at the operating system level. 4D includes a mechanism that automatically prevents the opening of a database when its data file, or one of its segments, is locked. In this case, when this detection option is activated, 4D displays a warning message and does not open the database:



Unless this option is checked, it is not possible to open a database when its data file is locked (default operation for 4D databases).

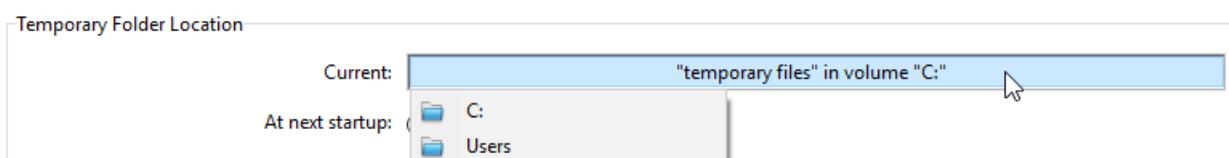
#### Sobre el bloqueo de archivos

Locked files can be read but their contents cannot be modified. For example, files are locked when they are stored on a non-rewritable support (DVD type) or when they are copied from this type of support. 4D can work in a transparent manner with locked data files, which allows, more particularly, the use of projects stored on DVD. However, this operation runs the risk of inadvertent use of a locked data file in which modifications will not be saved. This is why by default 4D does not allow databases with a locked data file to be opened.

### Ubicación de carpeta temporal

This area lets you change the location of temporary files created while 4D is running. The temporary files folder is used by the application, when necessary, to temporarily save the data in memory to disk.

The current location of this folder is displayed in the "Current:" area. You can click in this area to show the pathname as a scrolldown list:



Se ofrecen tres opciones de ubicación:

- System: When this option is selected, the 4D temporary files are created in a folder placed at the location specified by Windows and/or macOS. You can find out the current location defined by your system using the [Temporary folder](#) 4D command. The files are put into a subfolder whose name consists of the database name and a unique identifier.
- Data File Folder (default option): When this option is selected, the 4D temporary files are created in a folder named "temporary files" located at the same level as the data file of the database.

- User Defined: This option is used to set a custom location. If the location option is modified, it will be necessary to restart the database in order for the new option to be taken into account. 4D checks whether the folder selected can be write-accessed. If this is not the case, the application tries other options until a valid folder is found.

This option is stored in the "extra properties" of the structure that is available when the structure definition is exported in XML (see [Exporting and importing structure definitions](#)).

## Comparación de texto

If you change one of these options, you have to quit and reopen the database to make the change effective. Once the database is reopened, all of the database's indexes are automatically re-indexed.

- Consider @ as a wildcard only when at the beginning or end of text patterns : Allows you to set how the at sign "@" will be interpreted when used in a query or a character string comparison, when it is located in a word. When this option is not checked (default value), the at sign is used as the wildcard character, in other words, it replaces any character (see [Wildcard character \(@\)](#)).

When the option is checked, the at sign is regarded as a simple character if it is located within a word. This setting is especially useful when searching for E-mail addresses, where the @ sign is used internally. This option has an influence on searches, sorts, string comparisons, as well as on data stored in tables and data found in memory, like arrays. Fields and variables of the alpha (indexed or not) and text type are concerned by how the @ character is interpreted in searches and sorts.

Notas:

- For searches, if the search criteria begins or ends with @, the "@" character will be treated as a wildcard. Only if the "@" character is placed in the middle of a word (for example: [bill@cgi.com](mailto:bill@cgi.com)) does 4D treat it differently.
- This option can also have an influence on the behavior of the commands in the [Objects \(Forms\)](#) theme that accept the wildcard character ("@") in the object parameter.
- For security reasons, only the Administrator or Designer of the database can modify this parameter.
- Current data language: Used to configure the language used for character string processing and comparison. The language choice has a direct influence on the sorting and searching of text, as well as the character case, but it has no effect on the translation of texts or on the date, time or currency formats, which remain in the system language. Por defecto, 4D utiliza el lenguaje del sistema.

A 4D project can thus operate in a language different from that of the system. When a project is opened, the 4D engine detects the language used by the data file and provides it to the language (interpreter or compiled mode). Text comparisons, regardless of whether they are carried out by the project engine or the language, are done in the same language.

You can modify this setting in the application Preferences (see [General Page](#)). In this case, the setting applies to all the new databases created by 4D.

- Consider only non-alphanumeric chars for keywords : Modifies the algorithm used by 4D to identify keyword separators and hence build their indexes. By default, when this option is not checked, 4D uses a sophisticated algorithm that takes linguistic characteristics into account.

This algorithm is similar to the one used by word-processing software to determine the boundaries when selecting a word that is double-clicked. For more information about this algorithm, refer to the following address:  
<http://userguide.icu-project.org/boundaryanalysis>.

When this option is checked, 4D uses a simplified algorithm. In this configuration, any non-alphanumeric character (i.e., not a letter or a number) is considered as a keyword separator. This setting meets specific requirements associated with certain languages such as Japanese.

- Sorting order appropriate for searching : This option is only displayed when the Japanese language is selected. Modifies the interpretation of characters such as the "Katakana-Hiragana Prolonged Sound Mark" or "長音記号" or the "Japanese Iteration Marks" such as ">" or "়". Typical Japanese speaker is likely to prefer the results when the setting is enabled.

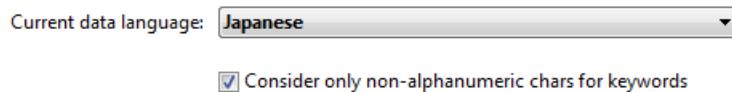
## Soporte de Mecab (versión japonesa)

On Japanese systems, 4D supports the *Mecab* library, with a indexing algorithm for keywords that is particularly suited for the Japanese language.

This algorithm is used by default in Japanese versions of 4D. Consider only non-alphanumeric chars for keywords : Modifies the algorithm used by 4D to identify keyword separators and hence build their indexes.

If needed, you can disable the use of the *Mecab* algorithm and use the conventional *ICU* library.

To disable *Mecab*, just check the Consider only non-alphanumeric chars for keywords option:



Note: You can also disable the use of Mecab by deleting or renaming the Resources/mecab folder of your 4D Japanese application.

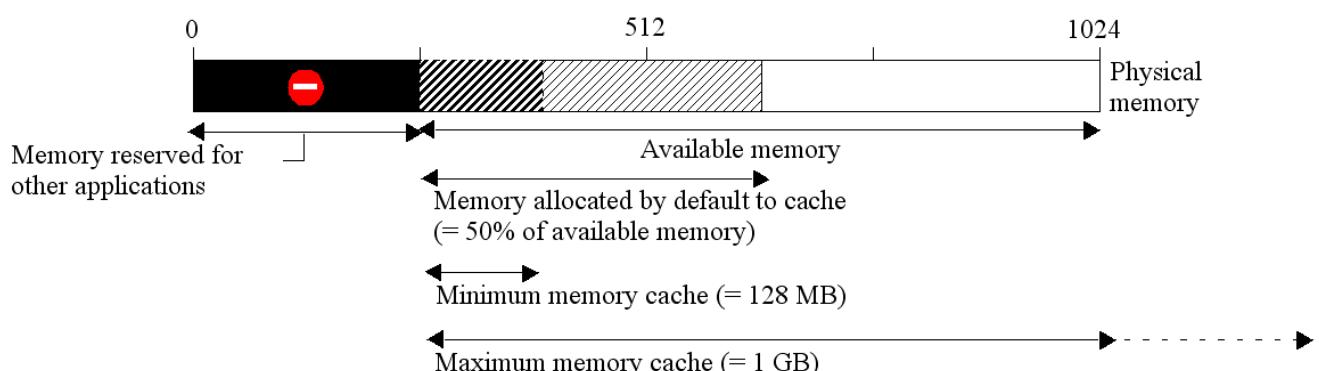
## Página Memoria

You use the settings on this tab to configure the cache memory for the database.

### Parámetros de la caché para la base

- Calculation of adaptive cache: When this option is checked, management of the memory cache is done dynamically by the system --- respecting limits that you set. This allows configuration of a high performance memory cache adapted to most configurations. The size of the memory cache is then calculated dynamically depending on set parameters. The values offered by default correspond to standard 4D usage.
  - Memory to be reserved for other applications and for the system : Portion of the RAM memory to reserve for the System and other applications. This value is increased for optimization when other applications are running on the same machine as 4D.
  - Percentage of available memory used for cache: Percentage of the remaining memory allocated to the cache by default.  
To obtain the size allocated by default to the cache, simply perform the following calculation: (Physical memory -- Physical memory to be reserved) X Percentage of the memory used for the cache. In the adaptive mode, the size of the memory cache varies dynamically depending on the needs of the application and the system. You can set limits using the following two options:
  - Minimum Size: Minimum amount of memory that must be reserved for the cache. Este valor no puede ser inferior a 100 MB.
  - Maximum Size: Maximum amount of memory that can be used by the cache. This value is virtually unlimited. Setting limits is particularly useful for databases that are distributed on machines for which you do not know the memory configuration a priori. In this case, the limits set let you guarantee a minimum performance in all cases. El siguiente diagrama ilustra este comportamiento:

Example for calculating cache memory: *Physical memory to reserve = 256 MB Percentage of the available memory used for the cache = 50% Maximum size = 1 GB Minimum size = 128 MB*



- Calculation of adaptive cache not checked: this mode, you set the size of the memory cache for the database yourself. 4D then displays an entry area that allows setting the memory cache to use as well as information related to the physical memory (RAM available on the machine), the current cache and cache after restart (taking your changes into account).

The size of the memory cache that you enter will be reserved for the 4D database, regardless of the state of machine resources. This setting can be used in certain specific configurations, or when the database is designed to be used on dissimilar systems in terms of memory. In most cases, the adaptive cache offers better performance.

- Flush Cache every... Seconds/Minutes: Specifies the time period between each automatic saving of the data cache, i.e., its writing to disk. 4D saves the data placed in the cache at regular intervals. You can specify any time interval between 1 second and 500 minutes. By default, 4D saves your data every 20 seconds. The application also saves your data to disk each time you change to another environment or exit the application. You can also call the [FLUSH CACHE](#) command to trigger the flush at any moment.

When you anticipate heavy data entry, consider setting a short time interval between saves. In case of a power failure, you will only lose the data entered since the previous save (if the database is running without a log file).

If there is a noticeable slowing down of the database each time the cache is flushed, you need to adjust the frequency. This slowness means that a huge amount of records is being saved. A shorter period between saves would therefore be more efficient since each save would involve fewer records and hence be faster.

By default, 4D displays a small window when the cache is flushed. If you do not want this visual reminder, you can uncheck the Flushing progress option on the [Interface page](#).

# Página Cliente/Servidor

The Client-server pages group together parameters related to the use of the database in client-server mode. Naturally, these settings are only taken into account when the database is used in remote mode.

## Página Opciones red

### Red

#### Publicar la base al inicio

This option lets you indicate whether or not the 4D Server database will appear in the list of published databases.

- When this option is checked (default), the database is made public and appears in the list of published databases (Available tab).
- When the option is not checked, the database is not made public and it does not appear in the list of published databases. To connect, users must manually enter the address of the database on the Custom tab of the connection dialog box.

If you modify this parameter, you must restart the server database in order for it to be taken into account.

#### Nombre de publicación

This option lets you change the publication name of a 4D Server database, *i.e.*, the name displayed on the dynamic Available tab of the connection dialog box (see the [Connecting to a 4D Server Database](#) section). By default, 4D Server uses the name of the project file. Puede introducir cualquier nombre personalizado que deseé.

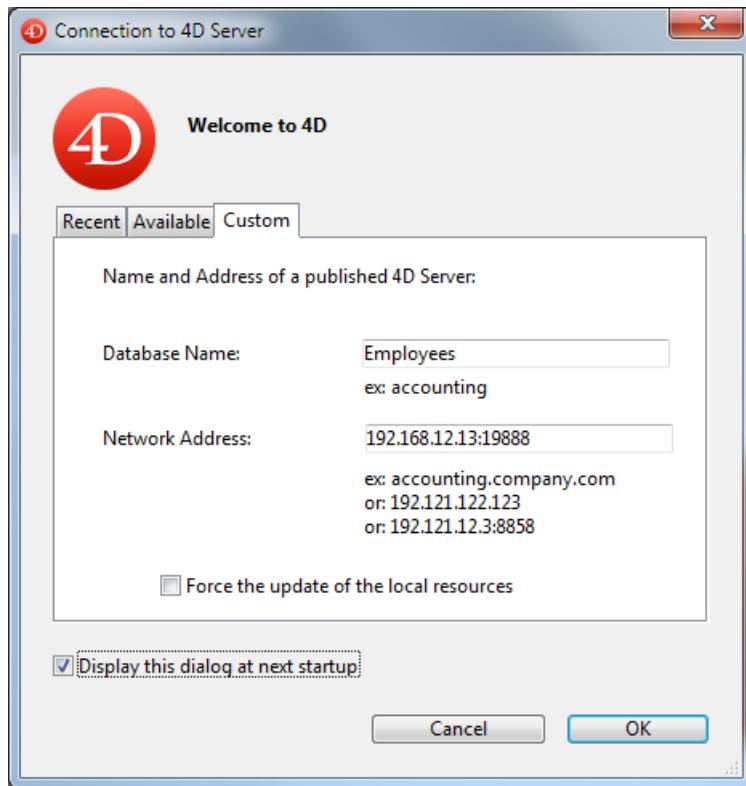
This parameter is not taken into account in custom client-server applications. In theory, the client application connects directly to the server application, without passing by the connection dialog box. However, in the event of an error, this dialog box can appear; in this case, the publication name of the server application is the name of the compiled project.

#### Número de puerto

This option lets you change the TCP port number on which 4D Server publishes the database. This information is stored in the project and on each client machine. By default, the TCP port number used by 4D Server and 4D in remote mode is 19813.

Customizing this value is necessary when you want to use several 4D applications on the same machine; in this case, you must specify a different port number for each application. When you modify this value from 4D Server or 4D, it is automatically passed on to all the 4D machines connected to the database.

To update any other client machines that are not connected, you just need to enter the new port number (preceded by a colon) after the IP address of the server machine on the Custom tab of the connection dialog box at the time of the next connection. Por ejemplo, si el nuevo número de puerto es 19888:



Only databases published on the same port as the one set in 4D client are visible on the TCP/IP dynamic publication page.

## 4D Server y números de puerto

4D Server uses three TCP ports for communications between internal servers and clients:

- SQL Server: 19812 by default (can be modified via the "SQL/Configuration" page of the Preferences).
- Application Server: 19813 by default (can be modified via the "Client-Server/Configuration" page of the Preferences, see above).
- DB4D Server (database server): 19814 by default . This port number cannot be modified directly but it always consists of the application server port number + 1.

When a 4D client connects to 4D Server, it uses the TCP port of the application server (19813 or the port indicated after the colon ':' in the IP address shown in the connection dialog box). Connection to other servers via their respective ports is then automatic; it is no longer necessary to specify them.

Note that in the case of access via a router or a firewall, the three TCP ports must be opened explicitly.

## Autenticación del usuario con el servidor de dominio

This option allows you to implement SSO (*Single Sign On*) capabilities in your 4D Server database on Windows. When you check this option, 4D transparently connects to the Active directory of the Windows domain server and gets the available authentication tokens. This option is described in the [Single Sign On \(SSO\) on Windows](#) section.

## Service Principal Name

When Single Sign On (SSO) is enabled (see above), you must fill in this field if you want to use Kerberos as your authentication protocol. This option is described in the [Single Sign On \(SSO\) on Windows](#) section.

## Tiempo de espera de conexiones Cliente-Servidor

This device is used to set the timeout (period of inactivity beyond which the connection is closed) between 4D Server and the client machines connecting to it. La opción ilimitada elimina el tiempo de espera. When this option is selected, client activity control is eliminated.

When a timeout is selected, the server will close the connection of a client if it does not receive any requests from the latter during the specified time limit.

## Comunicación cliente-servidor

### Registrar los clientes al Inicio para Execute On Client

When this option is checked, all the 4D remote machines connecting to the database can execute methods remotely. This mechanism is detailed in the section [Stored procedures on client machines](#).

### Cifrar las comunicaciones Cliente-Servidor

This option lets you activate the secured mode for communications between the server machine and the 4D remote machines. This option is detailed in the [Encrypting Client/Server Connections](#) section.

### Actualizar la carpeta Resources durante una sesión

This setting can be used to globally set the updating mode for the local instance of the Resources folder on the connected 4D machines when the Resources folder of the database is modified during the session (the Resources folder is automatically synchronized on the remote machine each time a session is opened). Hay tres parámetros disponibles:

- Never: The local Resources folder is not updated during the session. La notificación enviada por el servidor es ignorada. The local Resources folder may be updated manually using the Update Local Resources action menu command (see [Using the Resources explorer](#)).
- Always: The synchronization of the local Resources folder is automatically carried out during the session whenever notification is sent by the server.
- Ask: When the notification is sent by the server, a dialog box is displayed on the client machines, indicating the modification. Ask: When the notification is sent by the server, a dialog box is displayed on the client machines, indicating the modification. Automatic or manual mechanisms can be used to notify each client when the contents of this folder have been modified. For more information, please refer to the [Managing the Resources folder](#) section.

## Página Configuración IP

### Tabla de configuración Autorizar-Rechazar

This table allows you to set access control rules for the database depending on 4D remote machine IP addresses. This option allows reinforcing security, for example, for strategic applications.

This configuration table does not control Web connections.

The behavior of the configuration table is as follows:

- The "Allow-Deny" column allows selecting the type of rule to apply (Allow or Deny) using a pop-up menu. Para añadir una regla, haga clic en el botón Añadir. A new row appears in the table. The Delete button lets you remove the current row.
- The "IP Address" column allows setting the IP address(es) concerned by the rule. To specify an address, click in the column and enter the address in the following form: 123.45.67.89 (IPv4 format) or 2001:0DB8:0000:85A3:0000:0000:AC1F:8001 (IPv6 format). You can use an \* (asterisk) character to specify "starts with" type addresses. For example, 192.168.\* indicates all addresses starting with 192.168.
- The application of rules is based on the display order of the table. If two rules are contradictory, priority is given to the rule located highest in the table. You can re-order rows by modifying the current sort (click the header of the column to alternate the direction of the sort). También puede mover las líneas arrastrando y soltar.
- For security reasons, only addresses that actually match a rule will be allowed to connect. In other words, if the table only contains one or more Deny rules, all addresses will be refused because none will match at least one rule. If you want to deny only certain addresses (and allow others), add an Allow \* rule at the end of the table. Por ejemplo:
  - Deny 192.168.\* (deny all addresses beginning with 192.168)
  - Autorizar \* (y permitir todas las demás direcciones)

By default, no connection restrictions are applied by 4D Server: the first row of the table contains the Allow label and the \* (all addresses) character.



# Página Web

Using the tabs on the Web page, you can configure various aspects of the integrated Web server of 4D (security, startup, connections, Web services, etc.). For more information about how the 4D Web server works, see [Web server](#). For more information about 4D Web services, refer to the [Publication and use of Web Services](#) chapter.

## Configuración

### Información de publicación

#### Lanzar el servidor web al inicio

Indicates whether the Web server will be launched on startup of the 4D application. This option is described in the [Web server administration](#) section.

#### Activar HTTP

Indicates whether or not the Web server will accept non-secure connections. See [Enable HTTP](#).

#### Puerto HTTP

Número de puerto IP (TCP) de escucha para HTTP. See [HTTP Port](#).

#### Dirección IP

IP address on which the 4D web server will receive HTTP requests (4D local and 4D Server). See [IP Address to listen](#).

#### Activar HTTPS

Indicates whether or not the Web server will accept secure connections. See [Enable HTTPS](#).

#### Puerto HTTPS

Allows you to modify the TCP/IP port number used by the Web server for secured HTTP connections over TLS (HTTPS protocol). See [HTTPS Port](#).

#### Autorizar el acceso a la base de datos a través de las URL 4DSYNC

*Compatibility Note:* This option is [deprecated](#). For database access through HTTP, it is now recommended to use ORDA remote datastore features and REST requests.

## Rutas

### Raíz HTML por defecto

Define the default location of the Web site files and to indicate the hierarchical level on the disk above which the files will not be accessible. See [Root Folder](#).

### Página de inicio por defecto

Designa una página de inicio por defecto para el servidor web. See [Default Home page](#).

## Options (I)

### Caché

## Utilizar la caché Web de 4D

Activa la caché de la página web. See [Cache](#).

### Tamaño de la caché de las páginas

Define el tamaño de la caché. See [Cache](#).

### Vaciar la caché

En cualquier momento, puede vaciar la caché de las páginas y de las imágenes que contiene (si, por ejemplo, ha modificado una página estática y quiere volver a cargarla en la caché). En cualquier momento, puede vaciar la caché de las páginas y de las imágenes que contiene (si, por ejemplo, ha modificado una página estática y quiere volver a cargarla en la caché). La caché se borra inmediatamente.

You can also use the special URL </4DCACHECLEAR>.

## Procesos Web

This area allows you to configure how the web server will handle user sessions and their associated processes.

The Legacy sessions option is only available for compatibility in databases/projects created with 4D versions prior to 4D v18 R6.

### Sesiones extensibles (sesiones multiproceso)

When you select this option (recommended), a user session is managed through a Session object. See the [User sessions page](#).

### Sin sesiones

When this option is selected, the web server does not provide any specific support for [user sessions](#). Successive requests from web clients are always independent and no context is maintained on the server.

In this mode, you can configure additional web server settings:

- [Procesos Web simultáneos máximos](#)
- [Reuse Temporary Contexts \(4D in remote mode\)](#)
- [Utilizar los procesos apropiativos](#)

### Sesiones antiguas (sesiones procesos únicos)

*Compatibility Note:* This option is only available in databases/projects created with a 4D version prior to 4D v18 R6.

This option enables the handling of legacy user sessions by the 4D HTTP server. This mechanism is described in the [Web Sessions Management \(Legacy\)](#) section. See [Keep Session](#).

When selected, the [Reuse Temporary Contexts \(4D in remote mode\)](#) option is automatically checked (and locked).

### Procesos Web simultáneos máximos

Not available with [scalable sessions](#).

Strictly high limit of concurrent web processes. See [Maximum Concurrent Web Processes](#).

### Reutilización de los contextos temporales

Not available with [scalable sessions](#).

Allows you to optimize the operation of the 4D Web server in remote mode. See [Reuse temporary contexts in remote mode](#).

## Utilizar los procesos apropiativos

Not available with [scalable sessions](#).

Enables preemptive web processes in your compiled applications. When Use preemptive processes is selected, the eligibility of your web-related code (including 4D tags and web database methods) to the preemptive execution will be evaluated during the compilation. For more information, see [Using preemptive Web processes](#).

Esta opción no se aplica a las sesiones extensibles, a los procesos REST (modo compilado) ni a los procesos de servicios web (servidor o cliente). Ver [Activar el modo apropiativo para el servidor web](#).

## Tiempo de espera del proceso inactivo

Not available with [scalable sessions](#).

Le permite definir el tiempo de espera máximo antes de cerrar los procesos Web inactivos en el servidor. Ver [Duración de los procesos inactivos](#).

## Contraseñas Web

Define el sistema de autenticación que desea aplicar a su servidor web. Se proponen tres opciones:

Custom (default) Passwords with BASIC protocol  
Passwords with DIGEST protocol

Se recomienda utilizar la autenticación personalizada. Ver el capítulo [Autenticación](#) en la sección *Desarrollo Web*.

## Options (II)

### Conversión texto

Enviar directamente los caracteres extendidos

Ver [Propiedades obsoletas](#).

Standard Set

Define el conjunto de caracteres a utilizar por el servidor web 4D. Ver [Conjunto de caracteres](#).

### Conexiones Keep-Alive

Ver [Propiedades obsoletas](#).

### Parámetros CORS

Activar CORS

Activa el servicio Cross-origin resource sharing (CORS). Ver [Activar Cors](#).

Nombres de dominios/métodos HTTP permitidos

Lista de hosts y métodos permitidos para el servicio CORS. Ver [Parámetros CORS](#).

## Log (tipo)

### Formato del historial

Starts or stops the recording of requests received by the 4D web server in the *logweb.txt* file and sets its format. Ver [Registro de logs](#).

La activación y desactivación del archivo de historial de peticiones también se puede efectuar por programación utilizando el comando [WEB SET OPTION](#).

El menú de formato de registro ofrece las siguientes opciones:

- No Log File: When this option is selected, 4D will not generate a log file of requests.
- CLF (Common Log Format): When this option is selected, the log of requests is generated in CLF format. With the CLF format, each line of the file represents a request, such as:  
host rfc931 user [DD/MMM/YYYY:HH:MM:SS] "request" state length  
Each field is separated by a space and each line ends by the CR/LF sequence (character 13, character 10).
  - host: IP address of the client (ex. 192.100.100.10)
  - rfc931: information not generated by 4D, it's always - (a minus sign)
  - user: user name as it is authenticated, or else it is - (a minus sign). Si el nombre de usuario contiene espacios, se remplazan por \_ (un guión bajo).
  - DD: day, MMM: a 3-letter abbreviation for the month name (Jan, Feb,...), YYYY: year, HH: hour, MM: minutes, SS: seconds

La fecha y hora son locales al servidor.

- request: request sent by the client (ex. GET /index.htm HTTP/1.0)
- estado: respuesta dada por el servidor.
- length: size of the data returned (except the HTTP header) or 0.

Nota: por razones de rendimiento, las operaciones se guardan en una memoria búfer por paquetes de 1Kb antes de ser escritas en el disco. Las operaciones también se escriben en disco si no se ha enviado ninguna petición cada 5 segundos. Los valores posibles del estado son los siguientes: 200: OK 204: Sin contenido 302:

Redirección 304: No modificado 400: Petición incorrecta 401: Autenticación requerida 404: No encontrada 500: Error interno El formato CLF no puede ser personalizado.

- DLF (Combined Log Format): When this option is selected, the request log is generated in DLF format. DLF format is similar to CLF format and uses exactly the same structure. It simply adds two additional HTTP fields at the end of each request: Referer and User-agent.
  - Referer: Contains the URL of the page pointing to the requested document.
  - User-agent: Contains the name and version of the browser or software of the client at the origin of the request.

El formato DLF no se puede personalizar.

- ELF (Extended Log Format): When this option is selected, the request log is generated in ELF format. The ELF format is very widespread in the world of HTTP browsers. It can be used to build sophisticated logs that meet specific needs. For this reason, the ELF format can be customized: it is possible to choose the fields to be recorded as well as their order of insertion into the file.
- WLF (WebStar Log Format): When this option is selected, the request log is generated in WLF format. WLF format was developed specifically for the 4D WebSTAR server. It is similar to the ELF format, with only a few additional fields. Al igual que el formato ELF, se puede personalizar.

Configurar los campos Cuando selecciona el formato ELF (Extended Log Format) o WLF (WebStar Log Format), el área "Weg Log Token Selection" muestra los campos disponibles para el formato elegido. You will need to select each field to be included in the log. Deberá seleccionar cada campo para incluirlo en el registro.

Nota: no puede seleccionar el mismo campo dos veces.

The following table lists the fields available for each format (in alphabetical order) and describes its contents:

Campo	ELF	WLF	Valor
BYTES_RECEIVED		X	Número de bytes recibidos por el servidor
BYTES_SENT	X	X	Number of bytes sent by the server to the client
C_DNS	X	X	IP address of the DNS (ELF: field identical to the C_IP field)
C_IP	X	X	IP address of the client (for example 192.100.100.10)
CONNECTION_ID		X	Número de identificación de la conexión
CS(COOKIE)	X	X	Information about cookies contained in the HTTP request
CS(HOST)	X	X	Campo Host de la petición HTTP
CS(REFERER)	X	X	URL de la página que apunta al documento solicitado
CS(USER_AGENT)	X	X	Information about the software and operating system of the client
CS_SIP	X	X	Dirección IP del servidor
CS_URI	X	X	URI sobre el que se realiza la petición
CS_URI_QUERY	X	X	Parámetros de consulta de la petición
CS_URI_STEM	X	X	Parte de la petición sin los parámetros de consulta
DATE	X	X	DD: day, MMM: 3-letter abbreviation for month (Jan, Feb, etc.), YYYY: year
METHOD	X	X	HTTP method used for the request sent to the server
PATH_ARGS		X	CGI parameters: string located after the "\$" character
STATUS	X	X	Respuesta ofrecida por el servidor
TIME	X	X	HH: hour, MM: minutes, SS: seconds
TRANSFER_TIME	X	X	Tiempo solicitado por el servidor para generar la respuesta
USER	X	X	User name if authenticated; otherwise - (minus sign).
			If the user name contains spaces, they are replaced by _ (underlines)
URL		X	URL solicitado por el cliente

Las fechas y horas se indican en GMT.

## Historial (periodicidad)

Configure los parámetros de copia de seguridad automática para el registro de las peticiones. Primero debe elegir la frecuencia (días, semanas, etc.) o el criterio de límite de tamaño de los archivos haciendo clic en el botón de opción correspondiente. A continuación, debe especificar el momento preciso de la copia de seguridad si es necesario.

- No Backup: The scheduled backup function is deactivated.
- Every X hour(s): This option is used to program backups on an hourly basis. Puede introducir un valor entre 1 y 24
  - starting at: Used to set the time at which the first back up will begin.
- Every X day(s) at X: This option is used to program backups on a daily basis. Introduzca 1 si desea realizar una copia de seguridad diaria. When this option is checked, you must indicate the time when the backup must be started.
- Every X week(s), day at X: This option is used to program backups on a weekly basis. Introduzca 1 si desea realizar una copia de seguridad semanal. Introduzca 1 si desea realizar una copia de seguridad semanal. When this option is checked, you must indicate the day(s) of the week and the time when each backup must be started. You can select several days of the week if desired.

- Every X month(s), Xth day at X: This option is used to program backups on a monthly basis. Introduzca 1 si desea realizar una copia de seguridad mensual. Introduzca 1 si desea realizar una copia de seguridad mensual.
- Every X MB: This option is used to program backups based on the size of the current request log file. A backup is automatically triggered when the file reaches the set size. Puede definir un límite de tamaño de 1, 10, 100 o 1000 MB.

En el caso de las copias de seguridad programadas, si el servidor web no fue lanzado cuando se programó la copia de seguridad, en el siguiente lanzamiento 4D considera que la copia de seguridad ha fallado y aplica los parámetros adecuados, definidos en las Propiedades de la base.

## Web Services

Las opciones de esta pestaña permiten activar y configurar los servicios Web para el proyecto 4D, tanto por su publicación (lado del servidor) y su suscripción (lado del cliente).

Para más información sobre el soporte de los servicios web en 4D, consulte el capítulo [Publicación y uso de los servicios web](#).

### Servidor

Esta área contiene varias opciones relativas con el uso de 4D como un "servidor" de Servicios Web, es decir la publicación de los métodos proyecto en forma de Servicios Web.

- Allow Web Services Requests : This option lets you initialize the publication of Web Services. If this option has not been checked, 4D refuses SOAP requests and does not generate a WSDL - even if methods have the *Published in WSDL* attribute. When this option is checked, 4D creates the WSDL file.
- Web Service Name: This area lets you change the "generic name" of the Web Service. This name is used to differentiate the services both at the SOAP server level (when the server publishes several different Web Services), as well as in the Web Services directories. Por defecto, 4D utiliza el nombre A\_WebService.
- Web Services Namespace: This area is used to change the namespace of the Web Services published by 4D. Each Web Service published on the Internet must be unique. The uniqueness of the names of Web Services is ensured by using XML namespaces. A namespace is an arbitrary character string used to identify a set of XML tags in a unique way. Typically, the namespace begins with the URL of the company (<http://mycompany.com/mynamespace>). In this case, it is not indispensable to have anything in particular at the URL indicated; what matters is that the character string used is unique. By default, 4D uses the following namespace:  
<http://www.4d.com/namespace/default>.

Conforme al estándar XML para los nombres de etiquetas, las cadenas de caracteres utilizadas no deben contener espacios ni comenzar con un número. Además, para evitar cualquier riesgo de incompatibilidad, recomendamos que no utilice ningún carácter extendido (como los caracteres acentuados).

### Cliente

Esta área contiene varias opciones relacionadas con el uso de 4D como un "cliente" de Servicios Web, es decir, suscribirse a los servicios publicados en la red.

- Wizard Method Prefix: This area lets you change the prefix that is added automatically by 4D to the name of proxy methods generated by the Web Services Wizard. Proxy project methods form a link between the 4D application and the Web Services server. Por defecto, 4D utiliza el prefijo "proxy\_".

## Funcionalidades Web

Esta página contiene las opciones utilizadas para activar y controlar las funcionalidades web avanzadas, como el servidor REST.

### Publicación

## Activar el servicio REST

Inicia y detiene el servidor REST. Ver [Configuración del servidor REST](#).

## Acceso

Esta opción especifica un grupo de usuarios 4D que está autorizado a establecer la conexión a la base 4D utilizando las peticiones REST. Ver [Configuración del acceso REST](#).

## Web Studio

Enable access to the web studio

Activa el acceso general al studio web. Todavía hay que configurarlo en cada nivel del proyecto.

# Página SQL

This page is used to configure the publishing parameters, access rights, and engine options of the [4D SQL Server](#).

## Publicación del servidor SQL

See the [Configuration of 4D SQL Server](#) page on doc.4d.com.

## Control de acceso SQL para el esquema predeterminado

See the [Configuration of 4D SQL Server](#) page on doc.4d.com.

## Opciones del motor SQL

See the [SQL Engine Options](#) paragraph on doc.4d.com.

# Página PHP

In 4D, you can execute PHP scripts directly by configuring the PHP page of the Database Settings (see [Executing PHP scripts in 4D](#) in the 4D *Language Reference* manual).

## Intérprete

- IP Address and Port number By default, 4D provides a PHP interpreter, compiled in FastCGI. For reasons related to the internal architecture, execution requests go to the PHP interpreter at a specific HTTP address. By default, 4D uses the address 127.0.0.1 and port 8002. You can change this address and/or port if they are already used by another service or if you have several interpreters on the same machine. To do this, you modify the IP Address and Port number parameters. Note that the HTTP address must be on the same machine as 4D.
- External interpreter If you use an external PHP interpreter, it must be compiled in FastCGI and be on the same machine as 4D (see "Using another PHP interpreter or another php.ini file" in [Executing PHP scripts in 4D](#)). Select this option so 4D does not attempt a connection with the internal interpreter when executing a PHP request. Note that this configuration requires your manual execution and control of the external interpreter.

4D Server: These settings are shared between 4D Server and the 4D remote machines so it is not possible to use an external interpreter on the server machine and simultaneously use the internal interpreter on the client machines (and vice versa). Also, if the server uses an external interpreter on port 9002, the client machines must also use an interpreter on this port.

## Opciones

These options are related to the automatic management of the 4D PHP interpreter and are disabled when the External Interpreter option is selected.

- Number of processes: The 4D PHP interpreter drives a set of system execution processes called "child processes". For optimization, it can run and keep up to five child processes simultaneously by default. You can modify the number of child processes according to your needs. For example, you may want to increase this value if you call on the PHP interpreter intensively. For more information, refer to the "Architecture" section in [Executing PHP scripts in 4D](#).

Note: Under Mac OS, all child processes share the same port. Under Windows, each child process uses a specific port number. The first number is the one set for the PHP interpreter; the other child processes increment this number. For example, if the default port is 8002 and you launch 5 child processes, they will use ports 8002 to 8006.

- Restart the interpreter after X requests: This sets the maximum number of requests that the 4D PHP interpreter accepts. When this number is reached, the interpreter restarts. For more information about this parameter, refer to the FastCGI-PHP documentation.

Note: In this dialog box, the parameters are specified by default for all connected machines and all sessions. You can also modify and read them separately for each machine and each session using the [SET DATABASE PARAMETER](#) and [Get database parameter](#) commands. The parameters modified by the [SET DATABASE PARAMETER](#) command have priority for the current session.

# Página seguridad

This page contains options related to data access and protection for your desktop applications.

Note: For a general overview of 4D's security features, see the [4D Security guide](#).

## Acceso de usuarios remotos

These settings do not apply to project databases opened in single-user mode.

- Design and Runtime Explorer Access: Gives the specified group the ability to enter the Design environment of the database and display the Runtime Explorer.

Note que:

- Setting an access group in the Design environment also lets you deactivate the Create table option in the data import dialog box. For more information about this dialog box, refer to [Importing data from files](#).
- The Designer and Administrator always have access to the Design environment and Runtime Explorer, even if they are not explicitly part of the specified access group. For more information about users and user groups, refer to the [Managing 4D users and groups](#) chapter.
- Default User: When a Default User has been set, every user that opens the database or logs onto it has the same access privileges and restrictions defined for this Default User. Ya no es necesario ingresar un nombre de usuario. Moreover, if you have not associated a password with the Default User, the Password dialog box no longer appears and the database opens directly. This option simplifies access to the database while maintaining a complete data control system.
  - If you have associated a password with the Default User, a dialog box appears when the database is opened and the users must enter a password.¶
  - If you haven't associated a password with the Default User, the User Identification dialog box will not appear.

You can "force" the display of the User Identification dialog box when the "Default User" mode is active, for instance in order to connect as Administrator or Designer. To do so, press the Shift key while opening the database or connecting to it.

- Display User List in Password Dialog Box : If this option is checked, users must choose their name from the list of users and enter their password in the User Identification dialog box. If it is not checked, users must enter both their name and password. For more information about the two versions of the password dialog box, see the section "Access system overview" in [Access system overview](#).
  - User List in Alphabetical Order (only available if the previous option is checked): When this option is checked, the list of users in the password entry dialog box is sorted by alphabetical order.
- Users can change their password : When this option is checked, a Change button is displayed in the User Identification dialog box. This button lets the user access a dialog box that can be used to change their password (for more information about this dialog box, refer to the "Modification of password by user" in [Ensuring system maintenance](#)). If desired, you can hide the Change button so that users cannot modify their passwords. Para ello, desmarque esta opción.

## Opciones

- Filtering of commands and project methods in the formula editor and 4D Write Pro documents For security reasons, by default 4D restricts access to the commands, functions and project methods in the [Formula editor](#) in Application mode or added to multistyle areas or 4D Write Pro documents: only certain 4D functions and project methods that have been explicitly declared using the [SET ALLOWED METHODS](#) command can be used. You can completely or partially remove this filtering using the following options.

- Enabled for all (default option): Access to commands, functions and project methods is restricted for all users, including the Designer and the Administrator.
  - Disable for the Designer and the Administrator : This option grants full access to 4D commands and to methods only for the Designer and Administrator. It can be used to set up an unlimited access mode to commands and methods while remaining in control of the operations carried out. During the development phase, this mode can be used to freely test all the formulas, reports, and so on. During operation, it can be used to set up secure solutions that allow access to commands and methods on a temporary basis. This consists in changing the user (via the [CHANGE CURRENT USER](#) command) before calling a dialog box or starting a printing process that requires full access to the commands, then returning to the original user when the specific operation is completed. Note: If full access has been enabled using the previous option, this option will have no effect.
  - Disabled for all: This option disables control within formulas. When this option is checked, users have access to all the 4D commands and plug-ins as well as all project methods (except for invisible ones). Note: This option takes priority over the [SET ALLOWED METHODS](#) command. Cuando se selecciona, este comando no hace nada.
- Enable User Settings: You need to check this option to be able to display separated dialog boxes for user settings. When this option is checked, up to three dialog boxes are available: Structure Settings, User Settings, and User Settings for Data File. For more information, refer to [User settings](#).
  - Execute "On Host Database Event" method of the components: The [On Host Database Event database method](#) facilitates the initialization and backup phases for 4D components. For security reasons, you must explicitly authorize the execution of this method in each host database. Para hacer esto, debe marcar esta opción. By default, it is not checked.

Cuando esta opción está seleccionada:

- los componentes 4D están cargados,
- each [On Host Database Event database method](#) of the component (if any) is called by the host database,
- se ejecuta el código del método.

Cuando no está marcada:

- 4D components are loaded but they have to manage their initialization and backup phases themselves.
- the developer of the component has to publish the component methods that must be called by the host database during these phases (startup and shutdown)
- the developer of the host database must call the appropriate methods of the component at the right time (must be covered in the component documentation).

# Página de compatibilidad

The Compatibility page groups together parameters related to maintaining compatibility with previous versions of 4D.

The number of options displayed depends on the version of 4D with which the original database/project was created, as well as the settings modified in this database/project.

This page lists the compatibility options available for database/projects converted from 4D v18 onwards. For older compatibility options, refer to the [Compatibility page](#) on doc.4d.com.

- Use legacy network layer : Starting with 4D v15, 4D applications propose a new network layer, named `ServerNet`, to handle communications between 4D Server and remote 4D machines (clients). The former network layer has become obsolete, but it is kept to ensure compatibility with existing databases. Using this option, you can enable the former network layer at any time in your 4D Server applications depending on your needs. `ServerNet` is used automatically for new databases and databases converted from a v15 release or later. Note that in case of a modification, you need to restart the application for the change to be taken into account. Any client applications that were logged must also be restarted to be able to connect with the new network layer. Note: This option can also be managed by programming using the `SET DATABASE PARAMETER` command.
- Use standard XPath: By default this option is unchecked for databases converted from a 4D version prior to v18 R3, and checked for databases created with 4D v18 R3 and higher. Starting with v18 R3, the XPath implementation in 4D has been modified to be more compliant and to support more predicates. As a consequence, non-standard features of the previous implementation no longer work. Incluyen:
  - initial "/" is not the root node only - using a / as first character in a XPath expression does not declare an absolute path from the root node
  - no implicit current node - the current node has to be included in the XPath expression
  - no recursive searches in repeated structures - only the first element is parsed.¥
- Although not standard, you might want to keep using these features so that your code continues to work as before - - in this case, just set the option *unchecked*. On the other hand, if your code does not rely on the non-standard implementation and if you want to benefit from the extended XPath features in your databases (as described in the [DOM Find XML element](#) command), make sure the Use standard XPath option is *checked*.
- Use LF for end of line on macOS: Starting with 4D v19 R2 (and 4D v19 R3 for XML files), 4D writes text files with line feed (LF) as default end of line (EOL) character instead of CR (CRLF for xml SAX) on macOS in new projects. If you want to benefit from this new behavior on projects converted from previous 4D versions, check this option. See [TEXT TO DOCUMENT](#), [Document to text](#), and [XML SET OPTIONS](#).
- Don't add a BOM when writing a unicode text file by default: Starting with 4D v19 R2 (and 4D v19 R3 for XML files), 4D writes text files without a byte order mark (BOM) by default. In previous versions, text files were written with a BOM by default. Select this option if you want to enable the new behavior in converted projects. See [TEXT TO DOCUMENT](#), [Document to text](#), and [XML SET OPTIONS](#).
- Map NULL values to blank values unchecked by default a field creation : For better compliance with ORDA specifications, in databases created with 4D v19 R4 and higher the Map NULL values to blank values field property is unchecked by default when you create fields. You can apply this default behavior to your converted databases by checking this option (working with Null values is recommended since they are fully supported by [ORDA](#)).

# Página general

This page contains various options to configure the general operation of your 4D application.

## Opciones

### Al inicio

This option allows you to configure the default 4D display at startup, when the user launches only the application.

- Do nothing: Only the application window appears, empty.
- Open Local Project dialog: 4D displays a standard open document dialog box, allowing you to select a local project.
- Open last used project: 4D directly opens the last project used; no opening dialog box appears. >To force the display of the opening dialog box when this option is selected, hold down the Alt (Windows) or Option (macOS) key while launching the project.
- Open Remote Project dialog: 4D displays the standard 4D Server logon dialog, allowing you to select a project published on the network.
- Open Welcome Wizard dialog (factory setting): 4D displays the Welcome Wizard dialog box.

**4D Server:** The 4D Server application ignores this option. In this environment, the Do nothing mode is always used.

### Creación de formularios automática

This option is only used in binary databases; it is ignored in project architecture. Ver doc.4d.com.

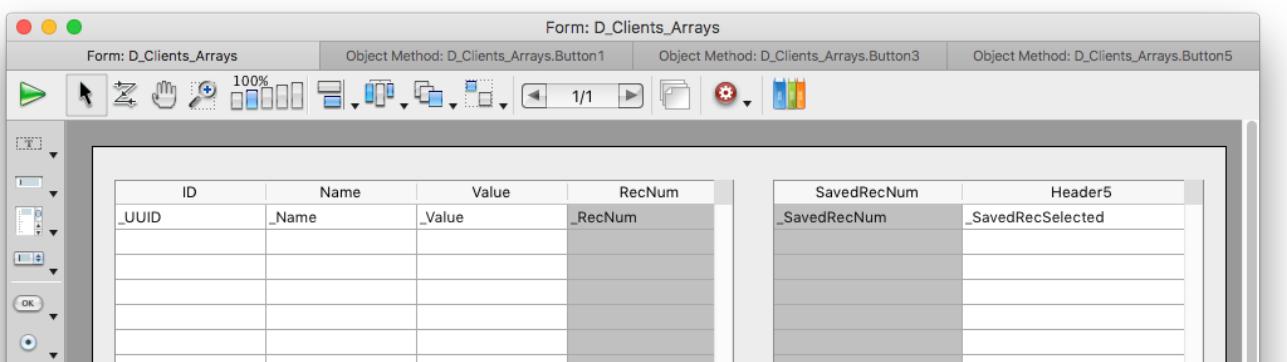
### Ventana con pestañas (sólo en macOS)

Starting with macOS Sierra, Mac applications can benefit from the Automatic Window Tabbing feature that helps organizing multiple windows: document windows are stacked into a single parent window and can be browsed through tabs. This feature is useful on small screens and/or when using a trackpad.

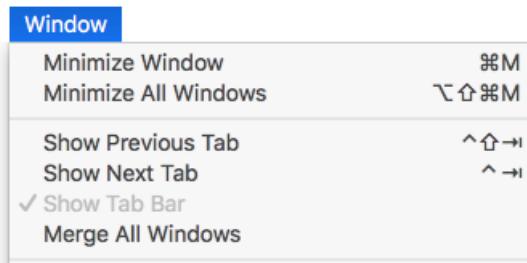
You can benefit from this feature in the following environments (with 4D 64-bit versions only):

- Ventana del editor de métodos
- Ventana del editor de formularios

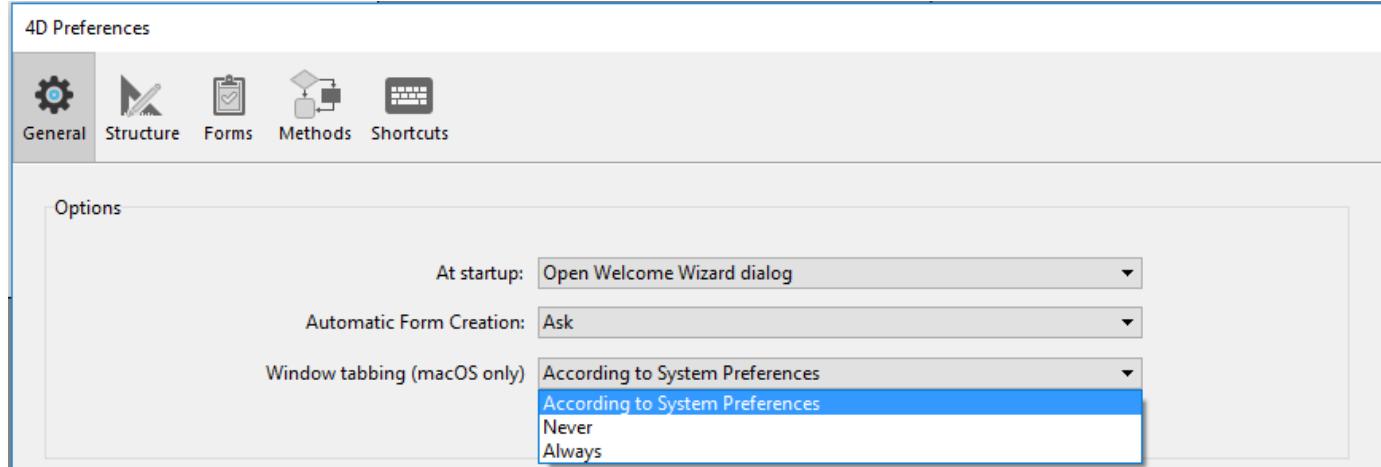
All windows from these editors can be put in tab form:



A set of commands in the Window menu allows managing the tabs:



In the 4D's Preferences dialog box, the Window tabbing option allows you to control this feature:



Hay tres opciones disponibles:

- According to System Preferences (default): 4D windows will behave like defined in the macOS System Preferences (In full screen, Always, or Manually).
- Never: Opening a new document in 4D form editor or method editor will always result in creating a new window (tabs are never created).
- Always: Opening a new document in 4D form editor or method editors will always result in creating a new tab.

## Apariencia (sólo para macOS)

This menu lets you select the color scheme to use for the 4D development environment. The specified scheme will be applied to all editors and windows of the Design mode.

You can also set the color scheme to use in your desktop applications in the "Interface" page of the Settings dialog box.

Hay tres opciones disponibles:

- According to System Color Scheme Preferences (default): Use the color scheme defined in the macOS System Preferences.
- Light: Use the Light Theme
- Sombra: utilizar el tema Sombra

Esta preferencia sólo es compatible en macOS. En Windows, se utiliza siempre el esquema "Light".

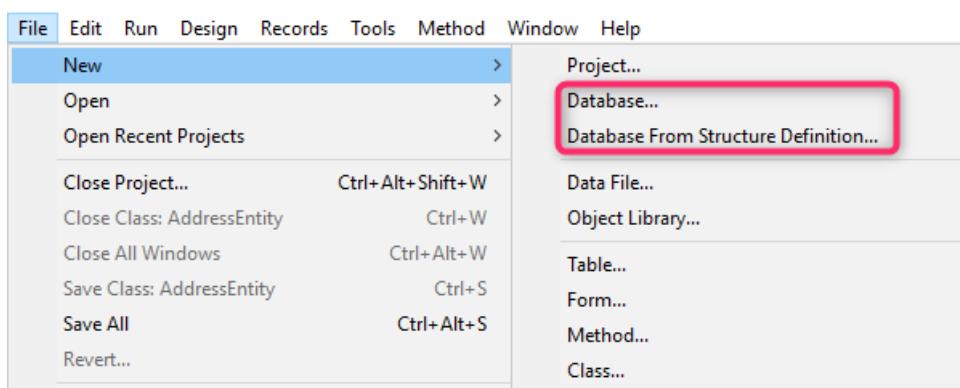
## Exit Design when going to Application Environment

If this option is checked, when the user switches to the Application environment using the Test Application menu command, all the windows of the Design environment are closed. If this option is not checked (factory setting), the windows of the Design environment remain visible in the background of the Application environment.

## Activar la creación de bases de datos binarias

If you check this option, two items are added in the File > New menu and the New toolbar button:

- Base de datos...
- Base de datos a partir de una definición de estructura...



These items allow you to create binary databases (see [Creating a new database](#) section). They are no longer proposed by default because 4D recommends using project-based architecture for new developments.

## When creating a new project

### Utilizar archivo historial

When this option is checked, a log file is automatically started and used when a new database is created. For more information, please refer to [Log file \(.journal\)](#).

### Crear un paquete

When this option is checked, 4D databases are automatically created in a folder suffixed .4dbase.

Thanks to this principle, under macOS the database folders appear as packages having specific properties. En Windows, este funcionamiento no tiene ningún impacto particular.

### Incluir los tokens en los archivos fuente del proyecto

When this option is checked, saved [method source files](#) in new 4D projects will contain tokens for classic language and database objects (constants, commands, tables and fields). Tokens are additional characters such as :C10 or :5 inserted in the source code files, that allow renaming tables and fields and identifying elements whatever the 4D version (see [Using tokens in formulas](#)).

If you intend to use VCS or external code editors with your new projects, you might want to uncheck this option for a better readability of the code with these tools.

This option can only be applied to projects (binary databases always include tokens).

You can always get the code with tokens by calling `METHOD GET CODE` with 1 in the *option* parameter.

### Exclusión de los tokens en los proyectos existentes

You can configure your existing projects to save code without tokens by inserting the following key in the `<applicationName>.4DProject` file using a text editor:

```
"tokenizedText": false
```

This setting is only taken into account when methods are saved. Existing methods in your projects are left untouched, unless you resave them.

## Crear el archivo .gitignore

You might need or want git to ignore some files in your new projects.

You can set this preference by checking the Create .gitignore file option.

When a project is created in 4D and that box is checked, 4D creates a `.gitignore` file at the same level as the Project folder (see [Architecture of a Project](#)).

You can define the default contents of the `.gitignore` file by clicking the pencil icon. This will open the `.gitignore` configuration file in your text editor. The contents of this file will be used to generate the `.gitignore` files in your new projects.

The [official git documentation](#) is a great resource to understand how `.gitignore` files work.

## Lenguaje de comparación de texte

This parameter configures the default language used for character string processing and comparison in new databases. The language choice has a direct influence on the sorting and searching of text, as well as the character case, but it has no effect on the translation of texts or on the date, time or currency formats, which remain in the system language. By default (factory setting), 4D uses the current user language set in the system.

A 4D database can thus operate in a language different from that of the system. When a database is opened, the 4D engine detects the language used by the data file and provides it to the language (interpreter or compiled mode). Text comparisons, regardless of whether they are carried out by the database engine or the language, are done in the same language.

When creating a new data file, 4D uses the language previously set in this menu. When opening a data file that is not in the same language as the structure, the data file language is used and the language code is copied into the structure.

You can modify this parameter for the open database using the Database Settings (see [Text comparison](#)).

## Ubicación de la documentation

This area configures access to the 4D HTML documentation displayed in your current browser:

- When you hit the F1 key while the cursor is inserted in a 4D class function or command name in the Method editor;
- When you double-click on a 4D command in the Commands Page of the Explorer.

## Lenguaje de la documentación

Lenguaje de la documentación HTML a mostrar. You can select a documentation in a different language from the application language.

## Primero buscar en la carpeta local

This option is only taken into account for command documentation access (excluding class functions).

Sets where 4D will look for documentation pages.

- When checked (default), 4D first looks for the page in the local folder (see below). If it is found, 4D displays the page in the current browser. If it is found, 4D displays the page in the current browser. This makes it possible to access the documentation even when you are offline.
- If it is not found, 4D displays an error message in the browser. When not checked, 4D looks for the desired page directly in the on-line documentation Web site and displays it in the current browser.

## Carpeta local

This option is only taken into account for command documentation access (excluding class functions).

Indicates the location of the static HTML documentation. By default, this is the ¥Help¥Command¥language subfolder. You can view the location by clicking on the menu associated with the area. If this subfolder is not present, the location is shown in red.

You can modify this location as desired, for example if you want to display the documentation in a language different from that of the application. The static HTML documentation can be located on another volume, on a web server, etc. To designate a different location, click on the [...] button next to the entry area and choose a documentation root folder (folder corresponding to the language: `fr`, `en`, `es`, `de` or `ja`).

# Página Estructura

## Llave primaria

These options in the preferences modify the default name and type of the primary key fields that are added automatically by 4D when new tables are created or by means of the [Primary key manager](#).

Las siguientes opciones están disponibles:

- Name ("ID" by default): Sets the default name of primary key fields. You can use any name you want, as long as it respects the [4D naming rules](#).
- Type ([Longint](#) by default): Sets the default type of primary key fields. You can choose the UUID type. In this case, the primary key fields created by default are of the [Alpha type](#) and have the UUID Format and Auto UUID field properties checked.

## Editor de estructura

This group of options configures the display of the 4D Structure editor.

### Calidad gráfica de la estructura

This option varies the level of graphic detail in the Structure editor. By default, the quality is set to High. You can select Standard quality in order to give priority to display speed. The effect of this setting is mainly perceptible when using the zoom function (see the "Zoom" paragraph in [Structure editor](#)).

Cuando una carpeta está atenuada, su contenido es:

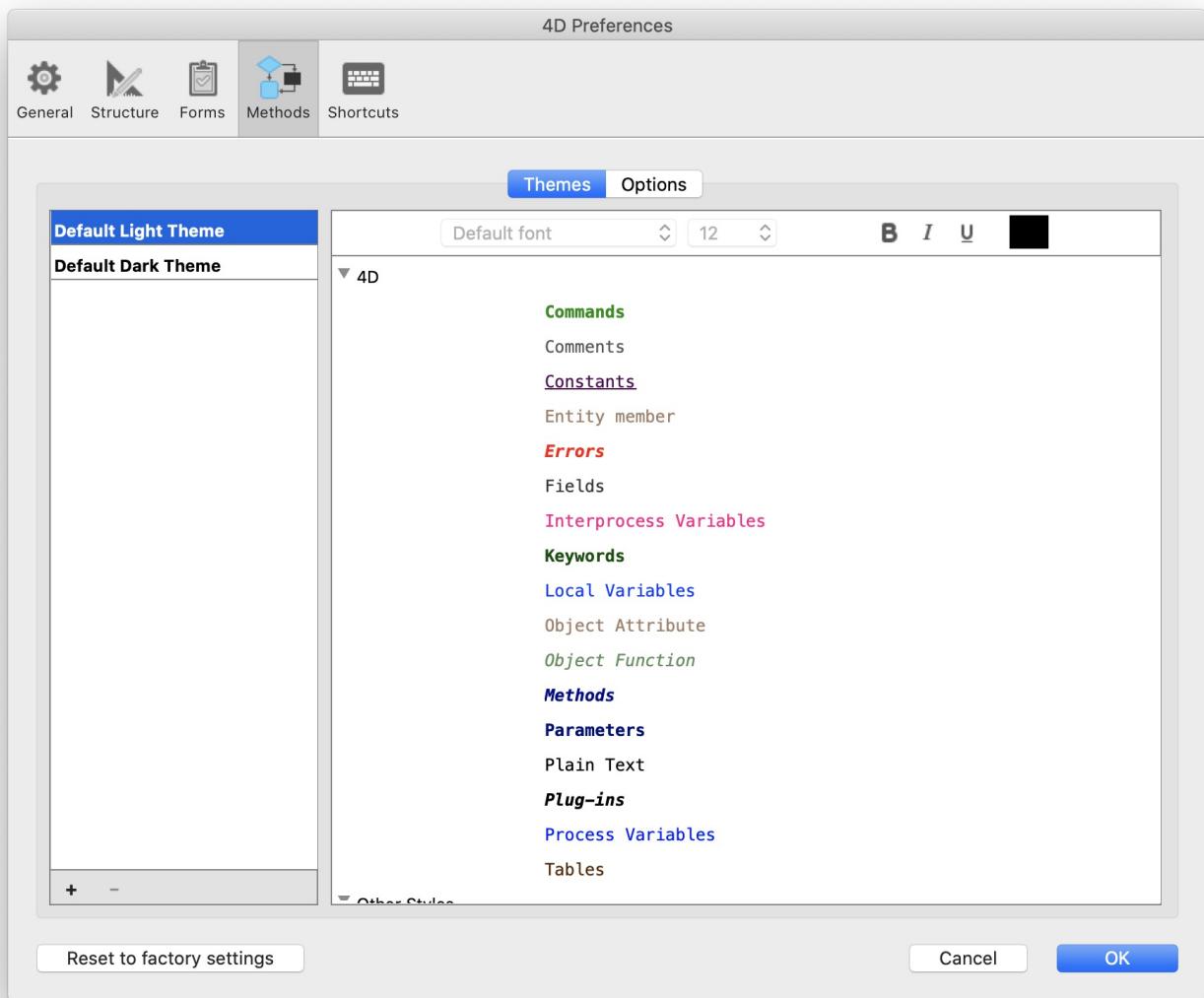
This option sets the appearance of dimmed tables in the Structure editor, when you carry out selections by folder (see [Highlight/dim tables by folder](#)). The possible options are Dimmed (a shadow replaces the table image) and Invisible (the table disappears completely).

# Página Métodos

This page contains parameters defining the Method editor interface and its default display as well as options concerning its operation. It is divided into two sections accessed using the Theme and Options tabs.

## Temas

This page allows selecting, creating, or configuring Method editor themes. A theme defines the font, font size, colors and styles of items displayed in the code editor.



## Lista de temas

In this list, you select the theme to apply to the code editor. All available themes are displayed, including custom themes (if any). 4D ofrece dos temas por defecto:

- Tema Light por defecto
- Tema oscuro por defecto

Los temas por defecto no pueden ser modificados ni eliminados.

A myTheme theme is automatically added if you already customized method editor styles in previous 4D releases.

## Creación de temas personalizados

You can create themes that you can fully customize. To create a theme, select an existing theme and click on the + at the bottom of the theme list. You can also add customized themes by copying theme files in the `4D Editor Themes` folder (see below).

## Archivos de temas personalizados

Each custom theme is stored in a single JSON file named `themeName.json`. The JSON files for custom themes are stored in the `4D Editor Themes` folder located at the same level as the 4D [preferences file](#).

If key values are not defined in a custom theme, they default to the values from the `Default Light Theme`. If a JSON theme file is invalid, the `Default Light Theme` is loaded and an error is generated.

When a theme file is modified by an external editor, 4D must be restarted to take the modification(s) into account.

## Definir el tema

Definir un tema significa:

- setting a global font and font size for the whole code editor,
- assigning specific styles and colors to each 4D language element (fields, tables, variables, parameters, SQL, etc.), SQL language element (keywords, functions, etc.), and color backgrounds.

Combining different colors and styles is particularly useful for code maintenance purposes.

## Fuentes y tamaños de fuente

The font and font size menus allows you to select the font name and size used in the Method editor entry area for all categories.

## Lenguaje 4D y lenguaje SQL

You can set different font styles and colors (font color or background color) for each type of language element. You can select the element(s) to customize in the Category list.

## Otros estilos

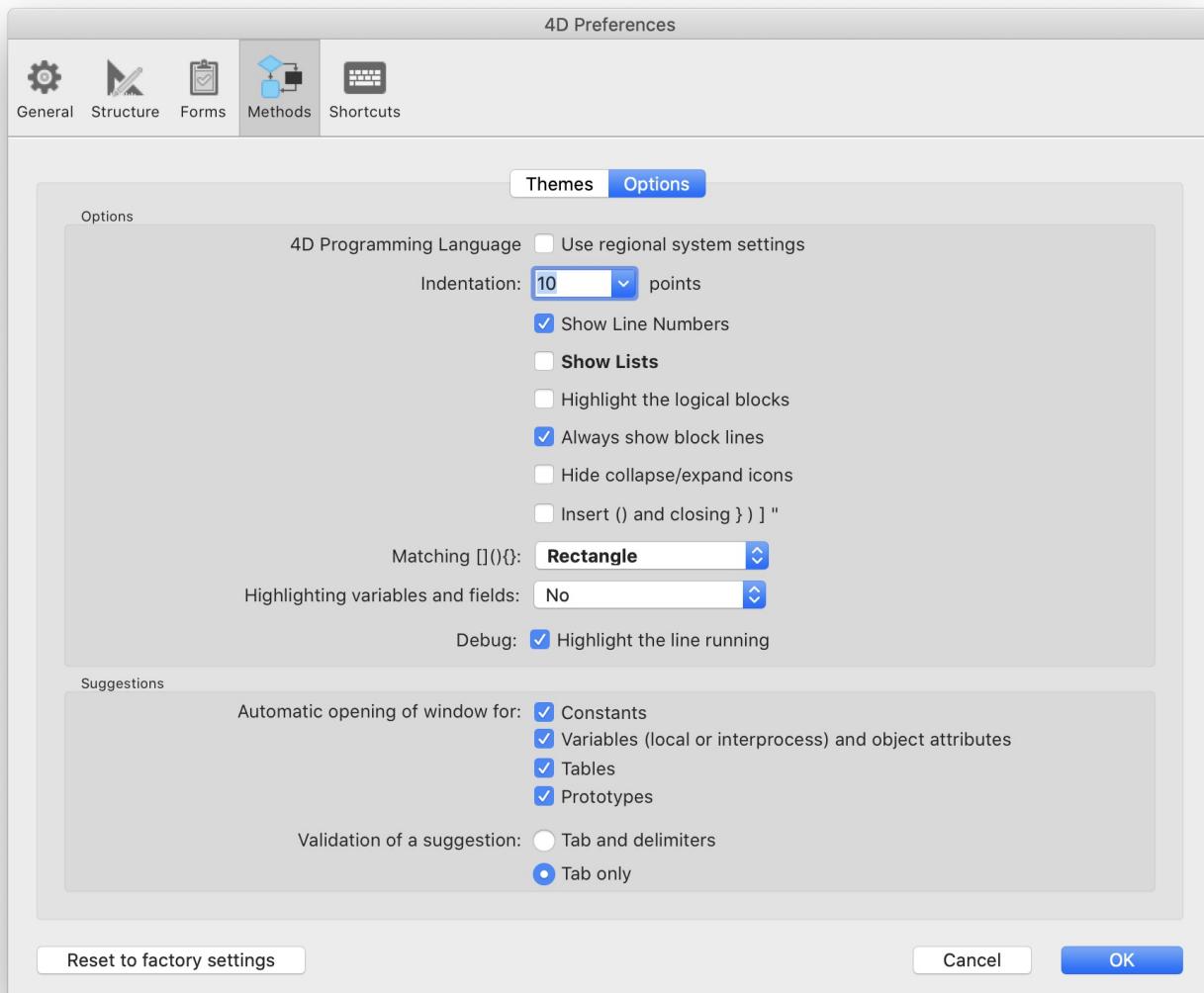
These options configure the various colors used in the Method editor and debugger interfaces.

Category	Element
› 4D Language	
▼ Other Styles	Background color
	Cursor line background color
	Highlight of the running line in the debugger
	Execution line background color
	Border of the running line in the debugger
	Highlight background color
	Highlight of the blocks
	Highlight of the parentheses
	Highlight of the found words
	Suggested text
	Selection back color
› SQL Language	

	Descripción
Color de fondo	Color de fondo de la ventana del editor de métodos.
Borde de la línea en ejecución en el depurador	Color of the border surrounding the line currently running in the debugger when the "Highlight line running" option is enabled in the <a href="#">Options</a> page.
Color de fondo de la línea del cursor	Color de fondo de la línea que contiene el cursor.
Color de fondo de la línea en ejecución	Background color of line being executed in the debugger.
Resaltar las palabras encontradas	Color de resaltado de las palabras encontradas en una búsqueda.
Destacar los paréntesis	Highlight color of corresponding parentheses (used when pairs of parentheses are signaled by highlighting, see <a href="#">Options</a> ).
Resaltado de los bloques	Highlight color for selected logical blocks when the "Highlight logical blocks" option is enabled in the <a href="#">Options</a> .
Resaltar la misma variable o campo	Highlight color for other occurrences of the same variable or field text when one of the "Highlighting variables and text" option is enabled in the <a href="#">Options</a> .
Resaltado de la línea en ejecución en el depurador	Highlight color of the line currently running in the debugger when the "Highlight line running" option is enabled in the <a href="#">Options</a> .
Color de fondo de la selección	Color de fondo de la selección.
Texto sugerido	Color of autocomplete text suggested by the Method editor.

## Opciones

This page configures Method editor display options.



## Opciones

### 4D Programming Language (Use regional system settings)

Allows you to disable/enable the "international" code settings for the local 4D application.

- unchecked (default): English-US settings and the English programming language are used in 4D methods.
- checked: Regional settings are used in 4D methods.

If you modify this option, you need to restart the 4D application so that the change is taken into account.

### Indentación

Changes the indentation value for the 4D code in the Method editor. The width must be specified in points (10 by default).

4D code is automatically indented in order to reveal its structure:

```

1  If ($vListItemPos#0)
2      // Get the list item information
3      GET LIST ITEM(hList;$vListItemPos;$v
4          // Is the item a Department item
5      If ($vListItemRef ?? 31)
6          // If so, it is a double-click
7          ALERT("You double-clicked on the
8      Else
9          // If not, it is a double-click
10         // Using the parent item ID to
11         $vDepartmentID:=List item parent
12         QUERY([Departments];[Departments]
13             // Tell where the Employee is
14             ALERT("You double-clicked on the
15         End if
16     End if
17
18
standard indentation

```

Modifying this default value can be useful if your methods contain complex algorithms with many levels of embedding. Narrower indentation can be used in order to limit horizontal scrolling.

### Mostrar los números de línea

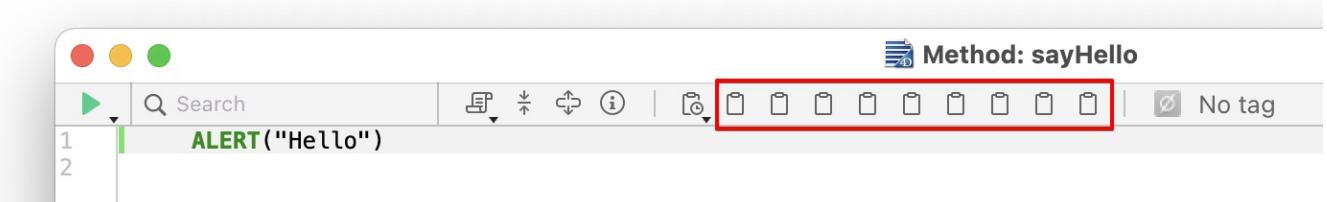
Lets you display the line numbers by default in each window of the Method editor. You can also show/hide line numbers for the current window directly from the Method editor.

### Mostrar las listas

Lets you choose whether or not to show the lists of objects (Commands, Tables and fields, etc.) by default when the Method editor window is opened. You can also show or hide each list directly from the Method editor.

### Mostrar los portapapeles

Lets you choose whether or not to show the multiple clipboards in the code editor.



The corresponding [clipboard shortcuts](#) are still active when these clipboards are hidden.

### Resaltado de los bloques lógicos

When checked, the whole code belonging to a logical block (If/End if for example) is highlighted when the mouse is placed over the expanded node:

```

12  If (<>PS_EditMovies=0)
13      <>PS_EditMovies:=New process($CurrentMethName;
14      Else
15          BRING TO FRONT(<>PS_EditMovies)
16      End if
17

```

The highlight color can be set in the [Theme](#) page.

### Mostrar siempre las líneas de bloques

Permite ocultar permanentemente las líneas verticales de bloques. The block lines are designed to visually connect nodes. By default, they are always displayed (except when collapse/expansion icons are hidden, see below).

9	└ If (Count	9	└ If (Count
10		10	
11	\$CurrentN	11	\$CurrentN
12	└ If (<>PS_	12	└ If (<>PS_
13	<>PS_Edi:	13	└ <>PS_Edi:
14	└ Else	14	└ Else
15	BRING_TC	15	BRING_TC
16	End if	16	End if
17		17	
18	└ Else	18	└ Else
19		19	

### Ocultar los iconos contraer/expandir

Allows you to hide all expand/collapse icons by default when displaying code. When the option is checked, node icons (as well as local block lines, see above), are displayed temporarily when the mouse is placed over a node:

9	If (Count	9	└ If (Count
10		10	
11	\$CurrentN	11	\$CurrentN
12	└ If (<>PS_	12	└ If (<>PS_
13	<>PS_Edi:	13	└ <>PS_Edi:
14	└ Else	14	└ Else
15	BRING_TC	15	BRING_TC
16	End if	16	End if
17		17	
18	└ Else	18	└ Else
19		19	

### Insertar () y añadir } ) ] " cierres

Enables automatic insertion of () and closing braces while typing code. Esta opción controla dos funcionalidades automáticas:

- parentheses pair (): Added after a 4D command, keyword or project method inserted from a suggestion or completion list, if the inserted element requires one or more mandatory arguments. For example, if you type "C\_OB" and press Tab, 4D writes "C\_OBJECT()" and sets the insertion point inside the ()..
- closing {}, ], or ": Character added when you type respectively an opening {, (, ], or ". This feature allows inserting matching pairs of symbols at the insertion point or surrounding a selected text. For example, if you highlight a string and type a single ", the whole selected string will be enclosed in ":".

The screenshot shows a code editor with the text "Hello\_world" selected. After pressing a key, the text is automatically enclosed in quotes, resulting in the selection becoming "Hello\_world".

### Correspondencia [](){}()

Sets the graphic signaling of matching braces in the code. This signaling appears whenever a square bracket, parenthesis, or curly bracket is selected. Las siguientes opciones están disponibles:

- Ninguno: sin señales
- Rectangle (default): Braces surrounded by a black line  
`INSERT MENU ITEM(main_bar;-1;Get indexed string(79;1);FileMenu)`
- Background Color: Braces highlighted (the color is set in the [Theme](#) page).
- Bold: Braces displayed in bold.

### Resaltado de las variables y campos

Allows to highlight all occurrences of the same variable or field in an open method window.

```

4   C_LONGINT(<>PS_EditMovies)
5   C_LONGINT($Window)
6   C_TEXT($CurrentMethodName)
7   C_LONGINT($Win)
8
9   If (Count parameters=0)
10
11  $CurrentMethodName:=Current method
12  If (<>PS_EditMovies=0)
13    <>PS_EditMovies:=New process ($C:
14  Else
15    BRING TO FRONT(<>PS_EditMovies)
16 End if

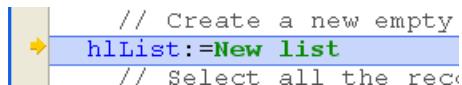
```

- No(default): No highlight
- On cursor: All occurrences are highlighted when the text is clicked
- On selection: All occurrences are highlighted when the text is selected

The highlight color can be set in the [Theme](#) page.

### Depurar (Resaltar la línea en ejecución)

Highlights the line that is currently running in the debugger in addition to the regular yellow arrow indicator.



```

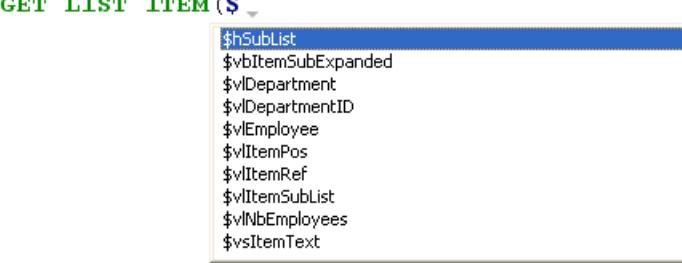
// Create a new empty
hList:=New list
// Select all the rec

```

If you deselect this option, only the yellow arrow is shown.

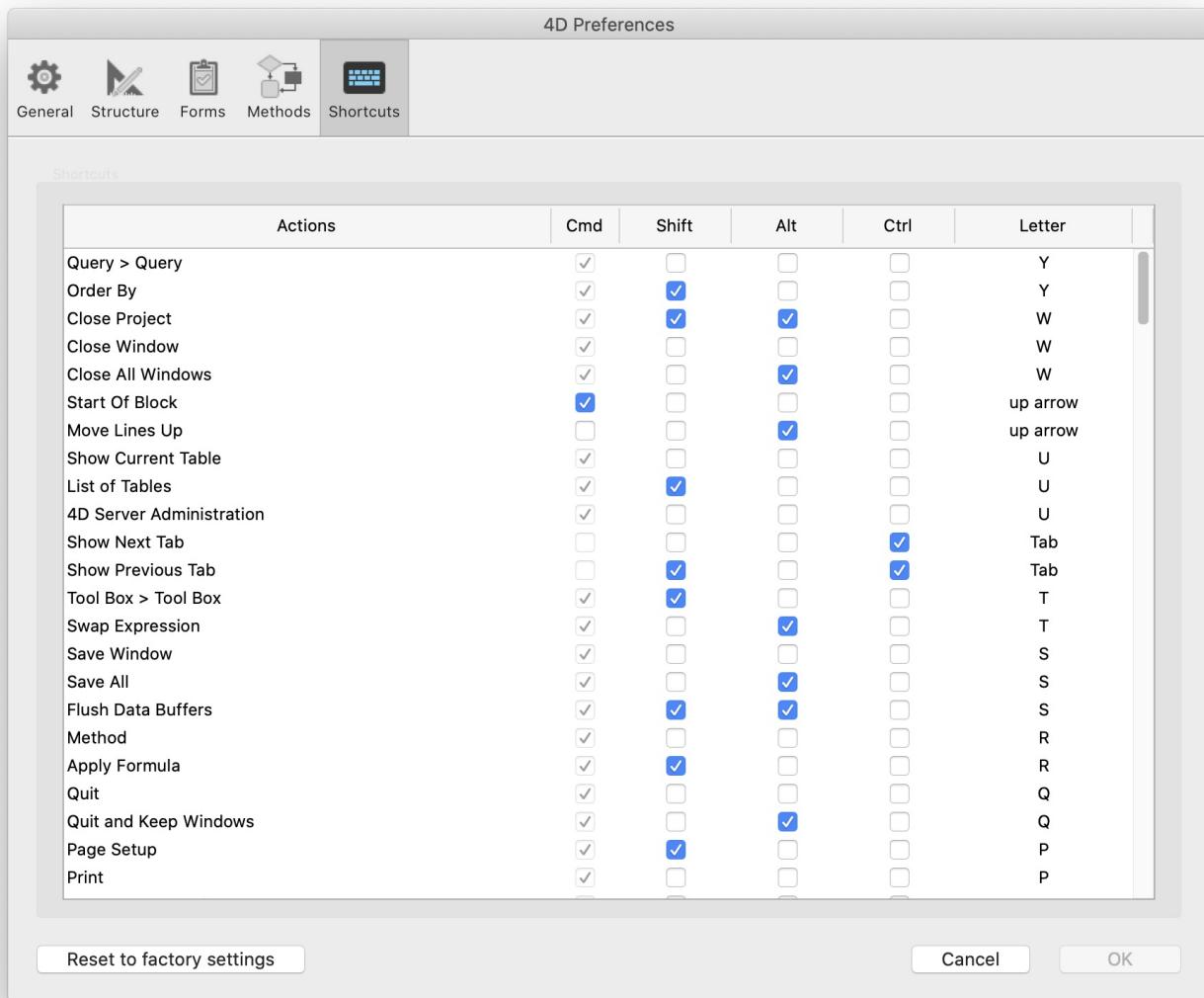
### Sugerencias

This area lets you configure autocomplete mechanisms in the Method editor to adapt it to your own work habits.

	<b>Descripción</b>
Apertura automática de la ventana	<p>Triggers the automatic display of the suggestion window for:</p> <ul style="list-style-type: none"> <li>• Constantes</li> <li>• Variables (local and interprocess) and object attributes</li> <li>• Tablas</li> <li>• Prototipos (es decir las funciones de clase)</li> </ul> <p>For example, when the "Variables (local or interprocess) and object attributes" option is checked, a list of suggestions appears when you type the \$ character:</p>  <p>You can disable this functioning for certain elements of the language by deselecting their corresponding option.</p>
Validación de una sugerencia	<p>Sets the entry context that allows the Method editor to validate automatically the current suggestion displayed in the autocomplete window.</p> <ul style="list-style-type: none"> <li>• Tab and delimiters</li> </ul> <p>When this option is selected, you can validate the current selection with the Tab key or any delimiter that is relevant to the context. For example, if you enter "ALE" and then "(", 4D automatically writes "ALERT(" in the editor. Here is the list of delimiters that are taken into account:</p> <p>( ; : = &lt; [ {</p> <ul style="list-style-type: none"> <li>• Tab only</li> </ul> <p>When this option is selected, you can only use the Tab key to insert the current suggestion. Esto se puede utilizar más concretamente para facilitar la introducción de caracteres delimitadores en los nombres de los elementos, como \${1}.</p> <p>Note: You can also double-click in the window or press the Carriage return key to validate a suggestion.</p>

# Página Atajos

This page lists all the shortcuts used in the 4D Design environment (except for standard "system" shortcuts, such as Ctrl+C/Command+C for the Copy command).



The screenshot shows the '4D Preferences' dialog box with the 'Shortcuts' tab selected. The main area is a table titled 'Shortcuts' with columns for 'Actions', 'Cmd', 'Shift', 'Alt', 'Ctrl', and 'Letter'. The 'Actions' column lists various commands, and the other columns show whether specific keyboard keys are assigned to them. At the bottom are buttons for 'Reset to factory settings', 'Cancel', and 'OK'.

Actions	Cmd	Shift	Alt	Ctrl	Letter
Query > Query	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Y
Order By	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Y
Close Project	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	W
Close Window	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	W
Close All Windows	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	W
Start Of Block	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	up arrow
Move Lines Up	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	up arrow
Show Current Table	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	U
List of Tables	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	U
4D Server Administration	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	U
Show Next Tab	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Tab
Show Previous Tab	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Tab
Tool Box > Tool Box	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	T
Swap Expression	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	T
Save Window	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	S
Save All	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	S
Flush Data Buffers	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	S
Method	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	R
Apply Formula	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	R
Quit	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Q
Quit and Keep Windows	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Q
Page Setup	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	P
Print	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	P

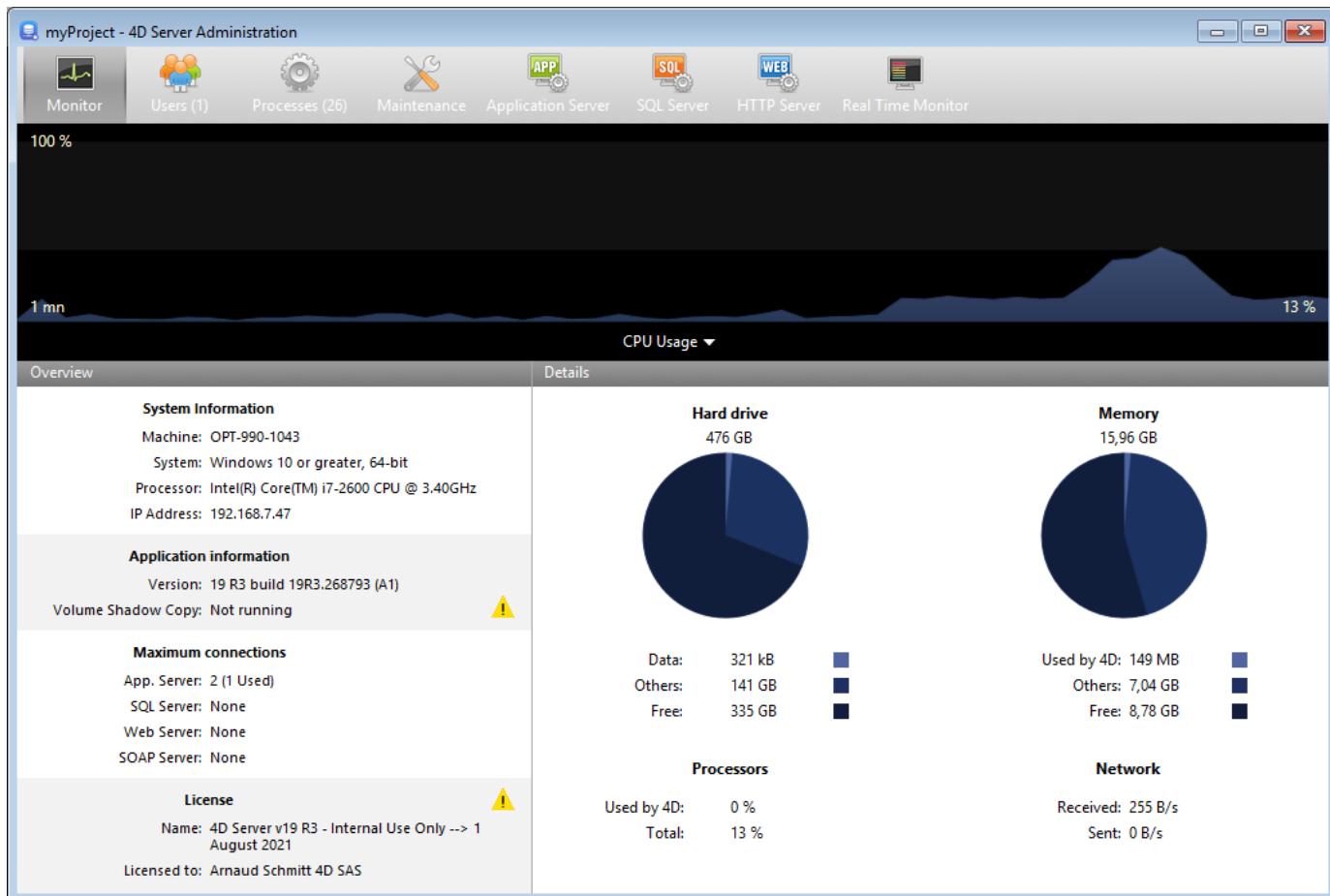
To modify a shortcut, you can select/deselect the item to modify (Shift, Alt or letter key) in the list. You can also double-click on a shortcut to configure it using a specific dialog box.

Note that each shortcut implicitly includes the Ctrl (Windows) or Command (macOS) key.

If you edit this list, your custom shortcuts settings are stored in a *4DShortcutsvXX.xml* file, created at the same level as the [user preferences file](#). Hence, each time 4D is updated your keyboard shortcut preferences remain.

# Página Monitor

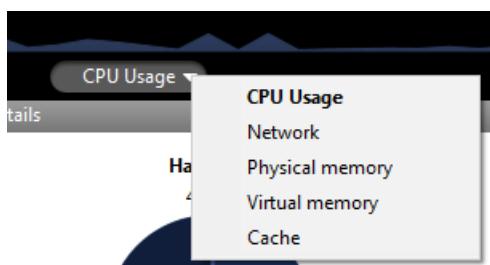
The Monitor page displays dynamic information concerning database use as well as information about the system and the 4D Server application.



On Windows, some of the system information displayed on this page are retrieved via the Windows "Performance Analyzer" tools. These tools can only be accessed when the user that opened the session where 4D Server was launched has the necessary administration authorization.

## Área gráfica

The graphic area lets you see the evolution in real time of several parameters: the CPU usage, network traffic and memory. You select the parameter to be displayed via a menu found in the center of the window:



- CPU Usage: Overall CPU usage of the machine, for all applications taken together. The specific part of 4D Server in this usage rate is provided in the "Processors" information area.
- Network: Number of bytes received per second by the machine (server or client). The number of bytes sent is provided in the "Network" information area.
- Physical memory: Quantity of RAM memory of machine used by 4D Server. A more detailed view of memory use is provided in the "Memory" information area.

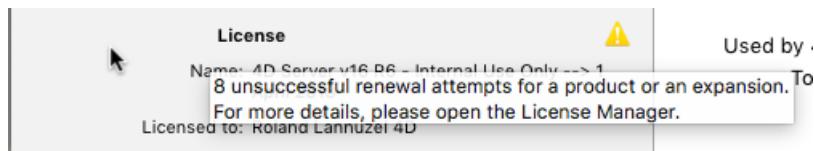
- Virtual memory: Quantity of virtual memory used by the 4D Server application. This memory is allocated by the system according to the application needs. The value found at the bottom right of the area indicates the quantity of memory currently being used. The value found at the top left indicates the maximum quantity of usable virtual memory. The maximum value is calculated dynamically according to the general memory settings of the application.
- Cache: Quantity of cache memory used by the 4D Server application. The value found at the bottom right of the area indicates the quantity of memory currently being used. The value found at the bottom right of the area indicates the quantity of memory currently being used.

Note that when this option is selected, the graph area scrolling is slowed down since an efficient analysis of the cache is generally carried out over a fairly long observation period.

## Visión general del área

The "Overview" area provides various information concerning the system, application and licenses installed on the 4D Server machine.

- System Information: Computer, system and IP address of server
- Application Information: Internal version number of 4D Server and Volume Shadow Copy status
- Maximum connections: Number of simultaneous connections allowed by type of server
- License: Description of license. When the product license or one of its attached expansions expires in less than 10 days, e.g. in case of a subscription-license, 4D Server tries to automatically renew the license from the 4D user account. In this case, if the automatic renewal failed for some reason (connection error, invalid account status, non-prolongated contract...), a warning icon is displayed next to the license to alert the server administrator. Additional information about the license renewal status can be displayed in a tip when you hover the mouse over the area:



Usually, you will need to check the [Licences Manager](#).

## Área de detalles

The "Details" area repeats part of the information displayed in the graphic area and provides additional information as well.

- Hard drive: Overall capacity of the hard disk and distribution of the space used by the database data (data file + data index), the space used by other files and the free space available.
- Memory: RAM memory installed on the machine and amount of memory used by 4D Server, by other applications or that is free. The memory used by 4D Server can also be displayed dynamically in the graphic area.
- Processors: Instant occupancy rate for processor(s) of the machine by 4D Server and by other applications. Esta tasa se recalculta constantemente. The occupancy rate by 4D Server can also be displayed dynamically in the graphic area.
- Network: Instantaneous number of bytes sent and received by the machine (server or client). Este valor se actualiza constantemente. The number of bytes received by can also be displayed dynamically in the graphic area.

# Página Usuarios

The Users page lists the 4D users connected to the server.

4D User	Machine name	Session name	IP Address	Login date	CPU Time	Activity
Designer	WIN7-ESMITH	esmith	localhost	03/05/2016 17:24	00:00:02	2%
Designer	MACWIN7	Arnaud	192.168.10.11	03/05/2016 17:27	00:00:01	0%

The "Users" button indicates, in parentheses, the total number of users connected to the server (this number does not take into account any display filters applied to the window). The page also contains a dynamic search area and control buttons. You can modify the order of the columns by dragging and dropping their header areas.

You can also sort the list of column values by clicking on its header. Click several times to specify in turn an ascending/descending order.



## Lista de usuarios

For each user connected to the server, the list provides the following information:

- System of the client machine (macOS or Windows) as an icon.
- 4D User: Name of the 4D user, or alias if set with the `SET USER ALIAS` command on the user machine. If passwords are not activated and no alias has been set, all users are named "Designer".
- Machine name: Name of the remote machine.
- Session name: Name of the session opened on the remote machine.
- IP Address: IP address of the remote machine.
- Login date: Date and time of the remote machine connection.
- CPU Time: CPU time consumed by this user since connecting.
- Activity: Ratio of time that 4D Server devotes to this user (dynamic display). "Sleeping" if the remote machine has switched to sleep mode (see below).

## Gestión de usuarios dormidos

4D Server specifically handles cases where a machine running a 4D remote application switches to sleep mode while its connection to the server machine is still active. In this case, the connected 4D remote application automatically notifies 4D Server of its imminent disconnection. On the server, the connected user changes to a Sleeping activity status:

4D User	Machine name	Session name	IP Address	Login date	CPU Time	Activity
Designer	macmini-program	program	192.168.18.11	20/10/15 09:43	00:00:18	Sleeping
Designer	iMac-VTalbot-0833	Vanessa Talbot	localhost	20/10/15 08:40	00:02:16	0%

Este estado libera recursos en el servidor. In addition, the 4D remote application reconnects to 4D Server automatically after waking up from sleep mode.

The following scenario is supported: a remote user stops working for awhile, for example during a lunch break, but keeps the connection to the server open. La máquina pasa al modo reposo. When the user returns, they wake the machine up and the 4D remote application automatically recovers its connection to the server as well as the session context.

A sleeping remote session is automatically dropped by the server after 48 hours of inactivity. You can modify this

default timeout using the [SET DATABASE PARAMETER](#) command with the [Remote connection sleep timeout](#) selector.

## Área de búsqueda/filtrado

This feature can be used to reduce the number of rows displayed in the list to those that correspond to the text entered in the search area. The area indicates the columns where the search/filtering will be carried out. On the Users page, it will be the 4D User, Machine name and Session name columns.

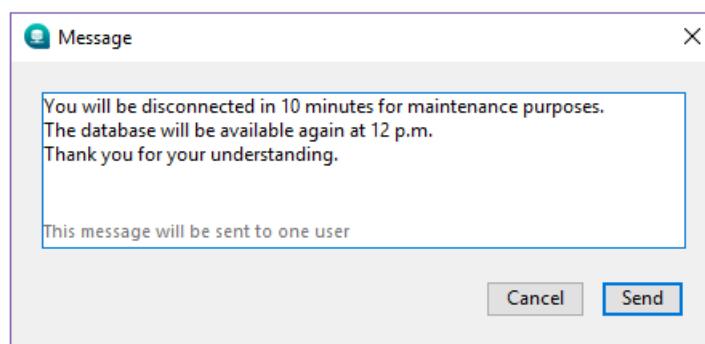
The list is updated in real time as you enter text in the area. It is possible to enter more than one value to be searched for: separate the values with a semi-colon. The `OR` type operator is used in this case. For example, if you enter "John;Mary;Peter," only rows with John OR Mary OR Peter in the target columns will be kept.

## Botones de administración

Esta página incluye tres botones de control. Estos botones están activos si se selecciona al menos una línea. You can select several rows by holding down the Shift key for an adjacent selection or Ctrl (Windows) / Command (macOS) key for a non-adjacent selection.

### Enviar mensaje

This button can be used to send a message to the 4D users selected in the window. Si no se selecciona ningún usuario, el botón no está activo. When you click on this button, a dialog box appears that lets you enter the message. The dialog box indicates the number of users that will receive this message:



The message will be displayed as an alert on the remote machines.

You can perform the same action for remote users with the [SEND MESSAGE TO REMOTE USER](#) command.

### Visualizar procesos

This button can be used to directly show the processes of the user(s) selected on the [Processes page](#) of the window. When you click on this button, 4D Server switches to the Processes page and enters the selected user names in the search/filtering area.

### Desconectar

This button can be used to force the selected user(s) to disconnect. When you click on this button, a warning dialog box appears so that you can confirm or cancel this operation (hold down Alt key while clicking on the Drop user button to disconnect the selected user(s) directly without displaying the confirmation dialog box).

You can perform the same action for remote users with the [DROP REMOTE USER](#) command.

# Página Procesos

The Processes page lists all the processes underway.

The screenshot shows the 'myProject - 4D Server Administration' window. The top navigation bar includes links for Monitor, Users (1), Processes (27) (which is the active tab), Maintenance, Application Server, SQL Server, WEB, and Real Time Monitor. A search bar at the top right contains the placeholder 'Session; Process name...'. Below the navigation is a table with the following columns: Process name, Session / Info, Type, Num v, State, CPU Time, and Activity. The table lists various processes including DB4D Cntriv, DB4D Flush, DB4D Index builder, DB4D Server, DB4D Sockets, Garbage Handler, HTTP Listener, Internal Timer Process, ServerNet select I/O handler, Task managers, TCP connection listener, User Interface, Application process, Design process, and Method9. The 'Method9' row is currently selected. At the bottom of the table, there is a note: 'Stored procedure - Method9 - Launched by aschmitt'. Below the table are five buttons: Abort Process, Pause Process, Activate Process, Debug Process, and Watch users. The status bar at the bottom right shows '88 %'.

The "Processes" button indicates, in parentheses, the total number of processes running in the server (this number does not take into account any display filters applied to the window nor the state of the Display processes by groups option).

You can change the order of the columns by simply dragging and dropping the column header areas. You can also sort the list of column values by clicking on its header.

Like the Users page, this page contains a dynamic [search/filtering area](#) that can be used to reduce the number of rows displayed in the list to those that correspond to the text entered in the search area. The search/filtering is carried out in the Session and Process name columns.

There are also three shortcut buttons that can be used to filter by the type of process displayed in the window:



- Users processes: Processes generated by and for the user sessions. These processes are preceded by an icon in the form of a figure.
- 4D Processes: Processes generated by the 4D Server engine. These processes are preceded by an icon in the form of a notched wheel.
- Spare processes: Processes that are inactive but kept temporarily and that can be reused at any time. This mechanism optimizes the reactivity of 4D Server. These processes are preceded by an icon in the form of a dimmed figure.

The Display processes by groups option lets you group together the internal processes of 4D Server as well as the client processes, for better readability. Cuando seleccione esta opción:

- the "twinned" 4D client processes (main 4D client process and 4D client base process, see [Process Type](#)) are grouped as one,

- a "Task managers" group is created; it includes the internal processes dedicated to dividing up tasks (Shared balancer, Net session manager, Exclusive pool worker),
- a "Client managers" group is created; it includes various client internal processes.

The lower area of the window is used to display the graphic representation of the activity of the selected process(es).

You can select several rows by holding down the Shift key for an adjacent selection or Ctrl (Windows) / Command (macOS) for a non-adjacent selection.

The activity of the process is the percentage of time that 4D Server has devoted to this process (ratio). The window provides the following information for each process:

- Tipo de proceso (ver abajo),
- Sesión/Información:
  - Proceso 4D - en blanco,
  - Proceso usuario - Nombre del usuario 4D,
  - Proceso web - ruta URL,
- Nombre del proceso,
- Number of the process (as returned by the `New process` command for example). The process number is the number assigned on the server. In the case of a global process, this number may be different from that assigned on the client machine.
- Estado actual del proceso,
- Running time (in seconds) of the process since its creation,
- Percentage of time that 4D Server has devoted to this process (ratio).

## Tipo del proceso

Each process is identified by an icon as well as a type. The color and form of the icon indicates the type of process:

icon	type
	Servidor de aplicación
	Servidor SQL
	Servidor DB4D (motor de base de datos)
	Servidor Web
	Servidor SOAP
	Protected 4D client process (development process of a connected 4D)
	Main 4D client process (main process of a connected 4D). Collaborative process, equivalent on the server of the process created on the client machine)
	4D client base process (process parallel to a 4D client process. Preemptive process responsible for controlling the corresponding main 4D client process)
	Spare process (former or future "4D client database process")
	Worker servidor SQL
	Proceso worker servidor HTTP
	4D client process (process running on the connected 4D)
	Stored procedure (process launched by a connected 4D and running on the server)
	Método web (lanzado por 4DACTION por ejemplo)
	Método web (apropiativo)
	Método SOAP (lanzado por un Web Service)
	Método SOAP (apropiativo)
	Logger (registrar)
	Listener conexión TCP
	Gestor de sesiones TCP
	Otro proceso
	Proceso worker (cooperativo)
	Proceso 4D client (apropiativo)
	Procedimiento almacenado (proceso apropiativo)
	Proceso worker (apropiativo)

Each main 4D client process and its "twinned" 4D client base process are grouped together when the `Display processes by groups` option is checked.

## Botones de administración

The page also has five control buttons that act on the selected process(es). Note that only user processes can be acted upon.



- Abort Process: can be used to abort the selected process(es). When you click on this button, a warning dialog box appears so that you can confirm or cancel the operation.

You can also abort the selected process(es) directly without displaying the confirmation dialog box by holding

down the Alt key while clicking on this button, or by using the [ABORT PROCESS BY ID](#) command.

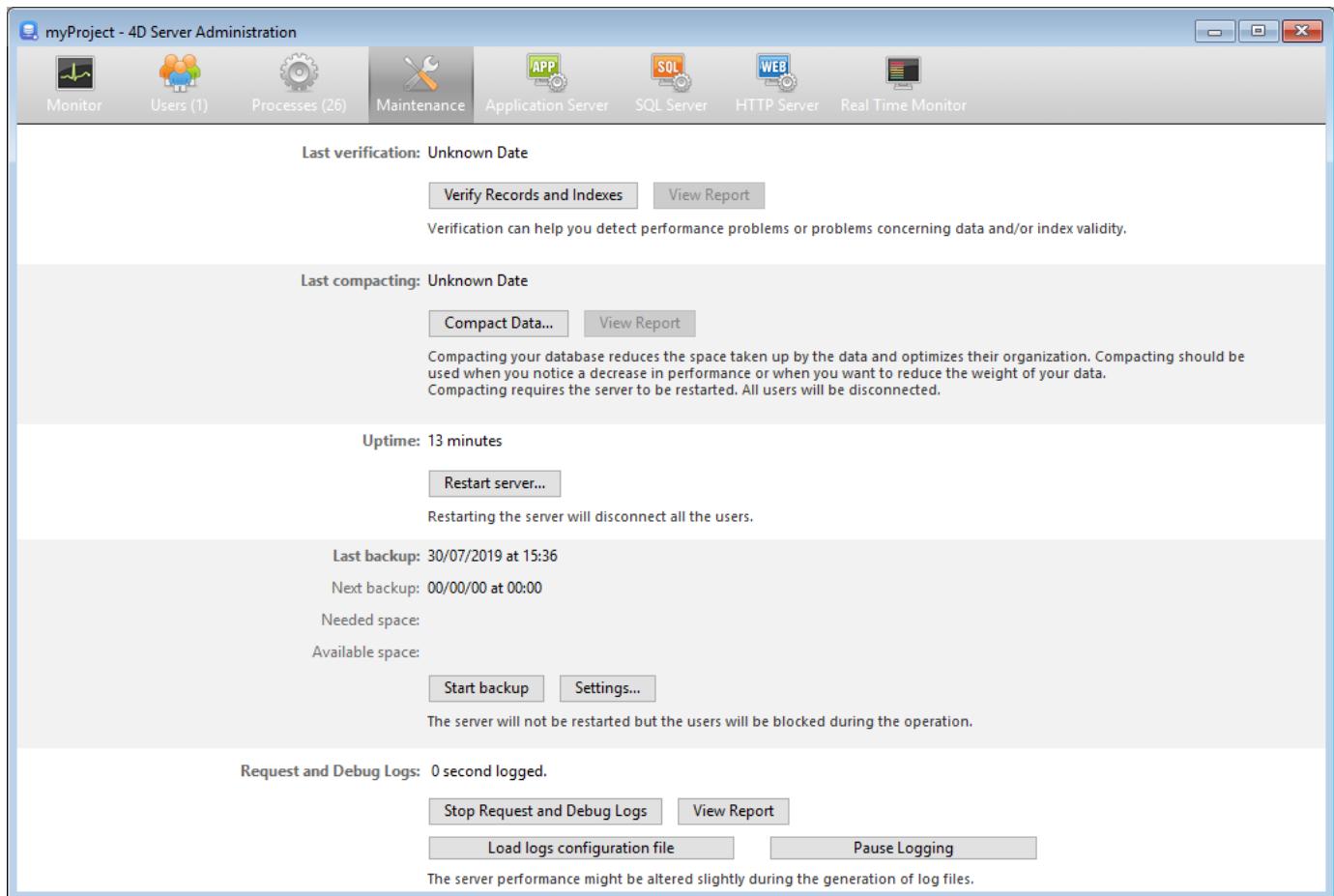
- Pause Process: can be used to pause the selected process(es).
- Activate Process: can be used to reactivate the selected process(es). The processes must have been paused previously (using the button above or by programming); otherwise, this button has no effect.
- Debug Process: can be used to open on the server machine one or more debugger windows for the selected process(es). When you click on this button, a warning dialog box appears so that you can confirm or cancel the operation. Note that the debugger window is only displayed when the 4D code is actually executed on the server machine (for example in a trigger or the execution of a method having the "Execute on Server" attribute).

You can also debug a process directly without displaying the confirmation dialog box by holding down the Alt key while clicking on this button.

- Watch users: used to display, on the [Users page](#), all the processes of the selected user(s). This button is active when at least one user process is selected.

# Página Mantenimiento

The Maintenance page of the 4D Server Administration window provides information concerning the current operation of the application. It also provides access to basic maintenance functions:



## Última verificación/compactación

These areas indicate the date, time and status of the last [data verification](#) and [compacting operation](#) carried out on the database.

### Verificar registros e índices

This button can be used to launch the verification operation directly, without interrupting the server. Note that the server may be noticeably slowed down during the operation.

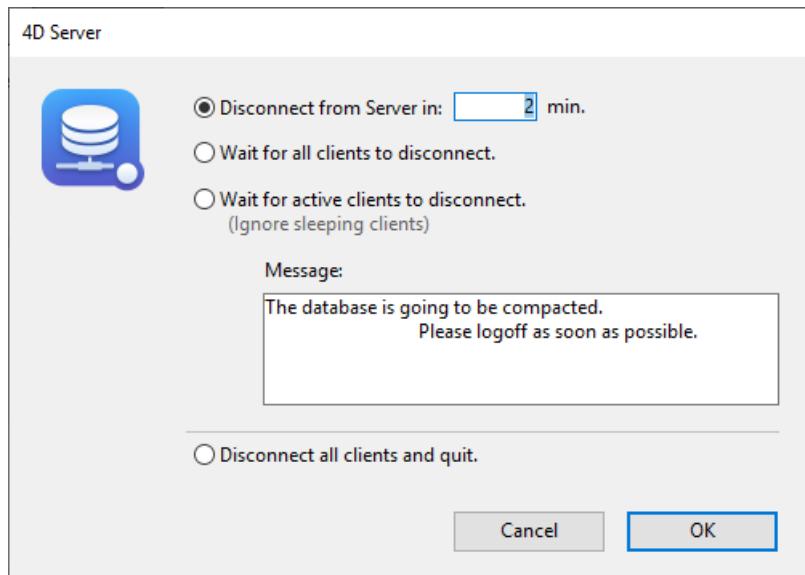
All the records and all the indexes of the database are verified. If you want to be able to target the verification or have additional options available, you will need to use the [Maintenance and Security Center](#) (MSC).

After verification, a report file is generated in XML format on the server in the [maintenance Logs](#) folder. The View Report button (named Download Report if the operation was carried out from a remote machine) lets you display the file in your browser.

This area indicates the date, time and status of the last carried out on the database data.

### Compactar los datos...

This button can be used to launch a data compacting operation directly. This operation requires stopping the server: when you click on this button, the 4D Server shutdown dialog box appears so that you can choose how to interrupt the operation:



After the actual interruption of the application service, 4D Server carries out a standard compacting operation on the database data. If you want to have additional options available, you will need to use the [MSC](#).

Once the compacting is finished, 4D Server automatically restarts the application. A continuación, los usuarios de 4D pueden volver a conectarse.

If the request for compacting was carried out from a remote 4D remote machine, this machine is automatically reconnected by 4D Server.

After verification, a report file is generated in XML format on the server in the [maintenance Logs](#) folder. The View Report button (named Download Report if the operation was carried out from a remote machine) lets you display the file in your browser.

## Tiempo de funcionamiento

This area indicates the duration of the 4D Server application execution since the last time it was started (days, hours and minutes).

### Reiniciar el servidor...

This button can be used to immediately close and restart the project. When you click on this button, the 4D Server shutdown dialog box appears so that you can choose how to interrupt the operation. After validation, 4D Server automatically closes and reopens the project. A continuación, los usuarios de 4D pueden volver a conectarse.

If the request for restarting was carried out from a remote 4D machine, this machine is automatically reconnected by 4D Server.

## Última copia de seguridad

This area indicates the date and time of the [last backup](#) of the database and provides information about the next scheduled automatic backup (if any). Automatic backups are configured using the Scheduler page of the structure settings.

- Last backup: date and time of last backup.
- Next backup: date and time of next scheduled backup.
- Needed space: estimated space needed for the backup. The actual size of the backup file may vary according to the settings (compression, etc.) and according to variations of the data file.
- Available space: space available on the backup volume.

The Start backup button can be used to backup the database immediately using the current backup parameters (files backed up, location of archives, options, etc.). You can view these parameters by clicking on the Settings... button.

During a backup on the server, the client machines are "blocked" (but not disconnected) and it is not possible for any new clients to connect.

## Historial de peticiones y depuración

This area indicates the server log files recording duration (when log files are activated) and allows you to control their activation.

Refer to the [Description of log files](#) section for details on log files.

### Iniciar/Detener Solicitud y Depurar Registros

The Start Request and Debug Logs button starts log files. Since this may noticeably deteriorate server performance, it is to be reserved for the development phase of the application.

This button only logs operations that are executed on the server.

When the logs have been activated, the button title changes to Stop Request and Debug Logs, so that you can stop recording requests at any time. Pay attention to the fact that restarting the log after stopping it "erases" the previous file.

### Ver el informe

The View Report button (named Download report if the operation was carried out from a remote desktop client) lets you open a system window displaying the request log file.

### Cargar archivo de configuración de logs

This button allows you to load a special server [log configuration file](#) ( .json file). Such a file can be provided by 4D technical services to monitor and study specific cases.

### Detener el registro

This button suspends all currently logging operations started on the server. This feature can be useful to temporarily lighten the server tasks.

When the logs have been paused, the button title changes to Resume logging, so that you can resume the logging operations.

You can pause and resume logging using the [SET DATABASE PARAMETER](#) command.

# Página Servidor de aplicación

The Application Server page groups together information about the desktop application published by 4D Server and can be used to manage this publication.

The screenshot shows the 'myProject - 4D Server Administration' window. The top navigation bar includes icons for Monitor, Users (1), Processes (26), Maintenance, Application Server (selected), SQL Server, HTTP Server, and Real Time Monitor. The Application Server section displays the following details:

- State:** Started
- Starting time:** 27/07/2021 at 18:29
- Uptime:** 1 day 15 hours 59 minutes
- Reject new connections** button (disabled)

**Configuration**

- Structure file: "myProject.4DProject" in volume "D:"
- Data file: "data.4DD" in volume "D:"
- Log file: data.journal
- Mode: Interpreted

**Launched as service:** No

- Listening to IP: 192.168.7.47
- Port: 19813
- TLS enabled: Yes

**Memory**

- Used cache memory: 733,88 kB
- Total cache memory: 400 MB

**Application Server Connections**

- Maximum: 2
- Used: 1

The upper part of the page provides information about the current status of the 4D Server application server.

- State: Started or Stopped.
- Starting time: Date and time the application server was launched. This date corresponds to the opening of the project by 4D Server.
- Uptime: Time elapsed since last opening of the project by the server.

## Aceptar/Rechazar nuevas conexiones

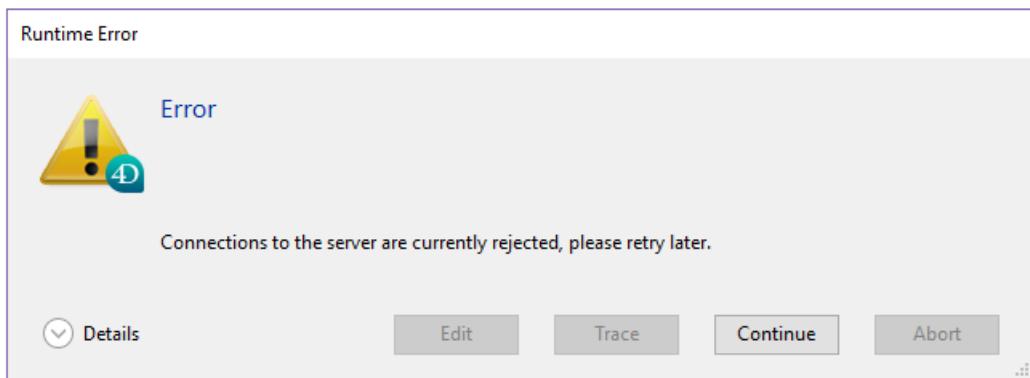
This button toggles and can be used to manage the access of new desktop client machines to the application server.

Por defecto, cuando se publica el proyecto:

- La etiqueta del botón es "Rechazar nuevas conexiones."
- New desktop clients can connect freely (within the limit of the connections permitted by the license).
- The project name is published in the remote connection dialog box (if the "At Startup Publish Database Name in the Connection Dialog" option is checked in the Preferences).

If you click on the Reject new connections button:

- The button title changes to "Accept new connections."
- Ningún nuevo cliente puede entonces conectarse. Clients attempting to connect will receive the following message:



- The project name no longer appears in the remote connection dialog box.
- Desktop clients that are already connected are not disconnected and can continue to work normally.

You can perform the same action with the [REJECT NEW REMOTE CONNECTIONS](#) command.

- If you click on the Accept new connections button, the application server returns to its default state.

This feature permits, for example, an administrator to carry out various maintenance operations (verification, compacting, etc.) just after having started the server. If the administrator uses a remote connection, they can be certain to be the only one modifying the data. It is also possible to use this function in preparation of a maintenance operation which requires that there be no desktop client machine connected.

## Information

### Configuración

This area provides information about the 4D project published by the server: name and location of data and structure files and name of database log file. You can click on the structure or data file name in order to view its complete pathname.

The Mode field indicates the current execution mode of the application: compiled or interpreted.

The lower part of the area indicates the server configuration parameters (launched as service, port and IP address) and the enabling of TLS for client-server connections (does not concern SQL nor HTTP connections).

### Memoria

This area indicates the Total cache memory (parameter set in the settings) and the Used cache memory (dynamic allocation by 4D Server according to its needs).

### Conexiones aplicación servidor

- Maximum: maximum number of simultaneous client connections allowed for the application server. This value depends on the license installed on the server machine.
- Used: actual number of connections currently being used.

# Página servidor SQL

The SQL Server page groups together information about the integrated SQL server of 4D Server. It also includes a button that can be used to control the activation of the server.

The screenshot shows the 'myProject - 4D Server Administration' window. The top navigation bar includes icons for Monitor, Users (1), Processes (26), Maintenance, Application Server, SQL Server (selected), HTTP Server, and Real Time Monitor. The main content area is titled 'SQL Server' and displays the following information:

- State:** Started
- Starting time:** 29/07/2021 at 10:30
- Uptime:** Less than one minute
- Configuration:** Auto-launched at startup: No, Listening to IP: 192.168.7.47, Listening on port: 19812, TLS enabled: No
- Connections:** Number of connections: 0
- Maximum connections:** SQL Server: None

A 'Stop SQL Server' button is located in the top right corner of the main content area.

The upper part of the page provides information about the current status of the SQL server of 4D Server.

- State: Started or Stopped
- Starting time: Date and time the SQL server was last launched.
- Uptime: Time elapsed since last startup of the SQL server.

## Iniciar/detener el servidor SQL

This button toggles and can be used to control the activation of the 4D Server SQL server.

- When the SQL server state is "Started," the button is titled Stop SQL Server. If you click on this button, the 4D Server SQL server is immediately stopped; it no longer replies to any external SQL requests received on the designated TCP port.
- When the SQL server state is "Stopped," the button is titled Start SQL Server. If you click on this button, the 4D Server SQL server is immediately started; it replies to any external SQL queries received on the designated TCP port. Note that you will need a suitable license to be able to use the 4D SQL server.

The SQL server can also be launched automatically on application startup (option in the Settings) or by programming.

## Información

### Configuración

This area provides information about the SQL server configuration parameters: automatic launching on startup, listening IP address, TCP port (19812 by default) and enabling of SSL for SQL connections (does not concern 4D nor HTTP connections).

These parameters can be modified via the 4D Settings.

## Conecciones

Number of SQL connections currently open on 4D Server.

## Conexiones máximas

Maximum number of simultaneous SQL connections allowed. This value depends on the license installed on the server machine.

# Página del servidor HTTP

The HTTP Server page groups together information about the operation of the Web server and SOAP server of 4D Server. The Web server lets you publish Web content such as HTML pages or pictures for Web browsers, and to handle REST requests. The SOAP server manages the publication of Web Services. These servers rely on the internal HTTP server of 4D Server.

The screenshot shows the 4D Server Administration interface with the title bar "myProject - 4D Server Administration". The top navigation bar includes icons for Monitor, Users (1), Processes (26), Maintenance, Application Server, SQL Server, WEB, and HTTP Server, with the HTTP Server icon being highlighted. Below the navigation bar, the main content area is divided into several sections:

- HTTP Server**: Shows the current state as "Started", starting time as "27/07/2021 at 18:29", uptime as "1 day 16 hours 3 minutes", and total HTTP hits as "0". A "Stop HTTP server" button is located in this section.
- Web information**: Shows web requests as "Accepted" and maximum connections as "None".
- SOAP information**: Shows SOAP requests as "Accepted" and maximum connections as "None". A "Reject SOAP requests" button is located in this section.
- HTTP server Configuration**: Provides detailed configuration settings:
  - Auto-launched at startup: Yes
  - HTTP Server processes (used/total): 5/7
  - Cache memory: 0 pages (0%)
  - Listening to IP: 192.168.7.47
  - HTTP port: 80
  - TLS enabled: Yes
  - HTTPS Port: 443
  - Log file: Logweb.txt
  - Log format: -
  - Next log backup: 00/00/00 at 00:00A "Clear cache" button is located in this section.

The upper part of the page provides information about the current status of the HTTP server of 4D Server.

- State: Started or Stopped
- Starting time: Date and time the HTTP server was last launched.
- Uptime: Time elapsed since last startup of the HTTP server.
- Total HTTP hits: Number of (low level) HTTP hits received by the HTTP server since it was started.

## Iniciar/detener el servidor HTTP

This button toggles and can be used to control the activation of the 4D Server HTTP server.

- When the HTTP server state is "Started," the button is titled Stop HTTP Server. If you click on this button, the 4D Server HTTP server is immediately stopped; the Web server, REST server, and SOAP server no longer accept any requests.
- When the HTTP server state is "Stopped," the button is titled Start HTTP Server. If you click on this button, the 4D Server HTTP server is immediately started; Web, REST, and SOAP requests are accepted.

You must have a suitable license in order to be able to start the HTTP server.

The HTTP server can also be launched automatically on application startup (Settings) or by programming.

## Información Web

This area provides specific information about the Web server of 4D Server.

- Web requests: Accepted or Rejected. This information indicates whether the Web server is activated. Since the Web server is directly linked to the HTTP server, Web requests are accepted when the HTTP server is started and rejected when it is stopped.
- Maximum connections: Maximum number of Web connections allowed. This value depends on the license installed on the server machine.

## Información SOAP

This area provides specific information about the SOAP server of 4D Server and includes a control button.

- SOAP requests: Accepted or Rejected. This information indicates whether the SOAP server is activated. In order for SOAP requests to be accepted, the HTTP server must be started and the SOAP server must explicitly accept the requests (see the Accept/Reject button).
- Maximum connections: Maximum number of SOAP connections allowed. This value depends on the license installed on the server machine.
- Accept/Reject SOAP requests button: This button toggles and can be used to control the activation of the 4D Server SOAP server. This button modifies the value of the Allow Web Services Requests option on the "Web Services" page of the Settings (and vice versa). You can also use the `SOAP REJECT NEW REQUESTS` command to refuse new SOAP requests, however this does not modify the value of the Allow Web Services Requests option.

If you click on the Accept SOAP requests button and the HTTP server is stopped, 4D automatically starts it.

## Configuración servidor HTTP

This area provides information about the configuration parameters and operation of the HTTP server:

- Auto-launched at startup: parameter set via the Settings.
- HTTP Server processes (used/total): number of HTTP processes created on the server (current number of processes / total of all processes created).
- Cache memory: size of HTTP server cache memory, when it is activated (size actually used by cache / maximum size theoretically allocated to the cache in the Settings). You can click on the Clear Cache button to empty the current cache.
- Listening to IP, HTTP Port (80 by default), TLS enabled for HTTP connections (does not concern 4D nor SQL connections) and HTTPS Port used: current [configuration parameters](#) of the HTTP server, specified through the Settings or by programming.
- Log file information: name, format and date of the next automatic log backup of the HTTP server (logweb.txt file).

# Página de seguimiento en tiempo real

The Real Time Monitor page monitors the progress of "long" operations performed by the application in real time. These operations are, for example, sequential queries, execution of formulas, etc.

Start Time	Duration (ms)	Information
2014-06-03 10:09:26.562	75 829	Sequential searching on Table_1: 4398275 of 24728607 records
2014-06-03 10:10:27.910	14 481	Deleting records: 41998 of 24728607

This page is available in the administration window of the server machine and also from a remote 4D machine. In the case of a remote machine, this page displays data from operations performed on the server machine.

A line is added for each long operation performed on the data. This line automatically disappears when the operation is complete (you can check the Display operations at least 5 seconds option to keep quick operations on screen for 5 seconds, see below).

The following information is provided for each line:

- Start Time: starting time of operation in the format: "dd/mm/yyyy - hh:mm:ss"
- Duration (ms): duration in milliseconds of operation in progress
- Information: title of operation.
- Details: this area displays detailed information which will vary according to the type of operation selected. En particular:
  - Created on: indicates whether the operation results from a client action (Created on client) or if it was started explicitly on the server by means of a stored procedure or the "Execute on server" option (Created on server).
  - Operation Details: Operation type and (for query operations) query plan.
  - Sub-operations (if any): Dependent operations of the selected operation (e.g. deleting related records before a parent record).
  - Process Details: Additional information concerning the table, field, process or client, depending on the type of operation

Real-time monitoring page uses the [GET ACTIVITY SNAPSHOT](#) command internally. More information can be found in this command description.

The page is active and updated permanently as soon as it is displayed. It should be noted that its operation can significantly slow the execution of the application. It is possible to suspend the updating of this page in one of the following ways:

- clicking on the Pause button,
- haciendo clic en la lista,
- presionando la barra espaciadora.

When you pause the page, a "PAUSED" message appears and the button label changes to Resume. You can resume monitoring of the operations by performing the same action as for pausing.

## Modo avanzado

The RTM page can display additional information, if necessary, for each listed operation.

To access the advanced mode for an operation, press Shift and select the desired operation. All available information is then displayed in the "Process Details" area without any filtering (as returned by the `GET ACTIVITY SNAPSHOT` command). Available information depends on the operation selected.

Here is an example of information displayed in standard mode:

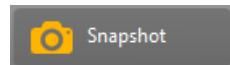
Start Time	Duration (ms)	Information	
2014-05-27 16:25:56	2 870	Sequential searching on Table_1: 438712 of 15128756 records	<b>Created on client</b>
			<b>Operation Details</b>
			Operation Type: <code>Query</code> Query Plan: "Table_1]Champ_2"="Table_1]Champ_2"
			<b>Process Details</b>
			Client Process Num: 7 Process Name: P_3 4D User: Super_Utilisateur Session Name: Arnaud Schmitt Machine Name: MACWIN7-SCHMITT

In advanced mode (Shift+Click on the operation), additional information is displayed:

Start Time	Duration (ms)	Information	
2014-05-27 16:25:56	2 870	Sequential searching on Table_1: 438712 of 15128756 records	<b>Created on client</b>
			<b>Operation Details</b>
			Operation Type: <code>Query</code> Query Plan: "Table_1]Champ_2"="Table_1]Champ_2"
			<b>Process Details</b>
			Client Process Num: 7 Process Name: P_3 4D User: Super_Utilisateur Session Name: Arnaud Schmitt Machine Name: MACWIN7-SCHMITT Client UID: B0C9071B0C9071B0C9071B0C90B0C9071 Client Version: v14 R2 Beta

## Botón Instantánea

The Snapshot button allows you to copy to the clipboard all the operations displayed in the RTM panel, as well as their related details (process and sub-operation info):



## Mostrar operaciones al menos 5 segundos

If you check the Display operations at least 5 seconds option, any listed operation will be displayed on the page for at least five seconds, even after its execution is finished. Retained operations appear dimmed in the operation list. This feature is useful for getting information about operations that execute very quickly.

# WebAdmin

Un componente de servidor web integrado, llamado `WebAdmin`, es utilizado por 4D y 4D Server para dar un acceso web seguro a funciones de gestión específicas como el [Explorador de datos](#). Puede conectarse local o remotamente a este servidor web desde un navegador o cualquier aplicación web y acceder a la aplicación 4D asociada.

El WebAdmin se encarga de la autenticación de los usuarios con privilegios "WebAdmin", para que puedan abrir sesiones de administración y acceder a las interfaces dedicadas.

Esta funcionalidad se puede utilizar en aplicaciones 4D que se ejecutan con o sin interfaces.

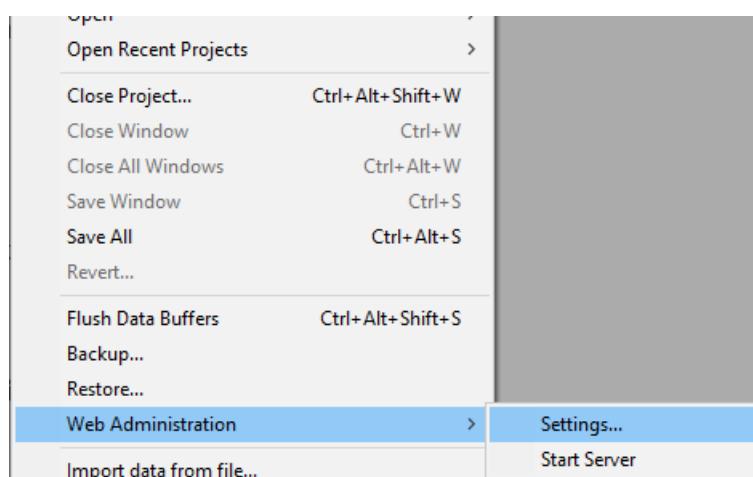
## Iniciar el servidor web WebAdmin

Por defecto, el servidor web `WebAdmin` no se lanza. Es necesario configurar el lanzamiento al inicio, o (en las versiones con interfaz) lanzarlo manualmente mediante una opción de menú.

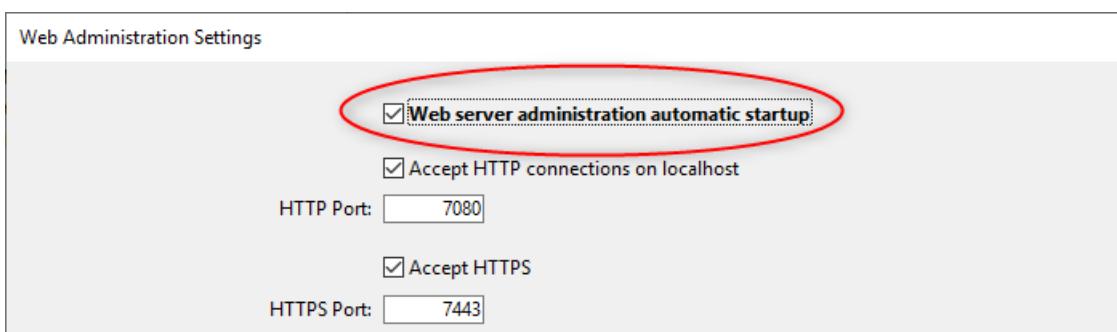
### Lanzamiento al inicio

Puede configurar el servidor web `WebAdmin` para que se lance al inicio de la aplicación 4D o 4D Server (antes de que se cargue cualquier proyecto).

- Si utiliza una aplicación 4D con interfaz, seleccione la opción de menú `Archivo > Administración web > Propiedades....`



Seleccione la opción Inicio automático de la administración del servidor web en la caja de diálogo de configuración:



- Tanto si utiliza la aplicación 4D con o sin interfaz, puede habilitar el modo de inicio automático utilizando el siguiente argumento *Interfaz de línea de comandos*:

```
open ~/Desktop/4D.app --webadmin=auto-start true
```

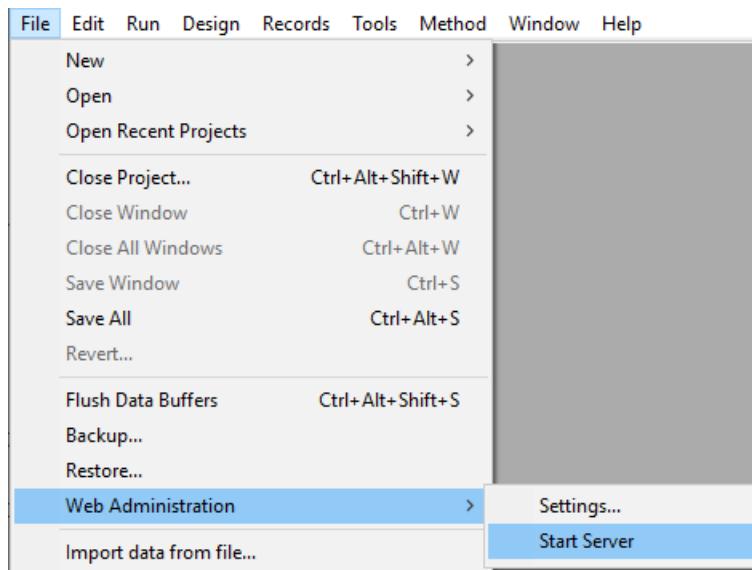
Si el puerto TCP utilizado por el servidor web `WebAdmin` ([HTTPS](#) o [HTTP](#), según la configuración) no está

disponible al inicio, 4D intentará sucesivamente los 20 puertos siguientes, y utilizará el primero que esté disponible. Si no hay ningún puerto disponible, el servidor web no se lanza y se muestra un error o para las aplicaciones sin interfaz, aparece en la consola.

## Iniciar y detener

Si utiliza una aplicación 4D con interfaz, puede iniciar o detener el servidor web `WebAdmin` de su proyecto en cualquier momento:

Seleccione la opción de menú Archivo > Administración web > Iniciar el servidor.



El elemento de menú se convierte en Detener el servidor cuando se lanza el servidor; seleccione Detener el servidor para detener el servidor web `WebAdmin`.

## Propiedades WebAdmin

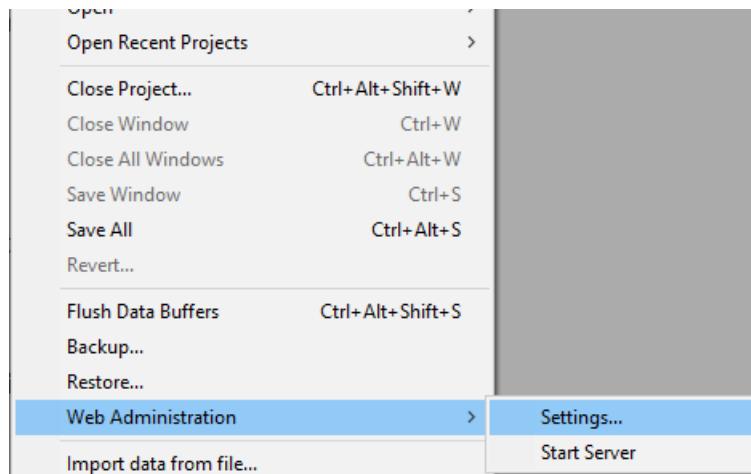
La configuración del componente `WebAdmin` es obligatoria, en particular para definir la **\*\* clave de acceso\*\***. Por defecto, cuando la clave de acceso no está configurada, no se permite el acceso a través de una URL.

Puede configurar el componente `WebAdmin` en la caja de diálogo **Configuración de la administración web** (ver más abajo).

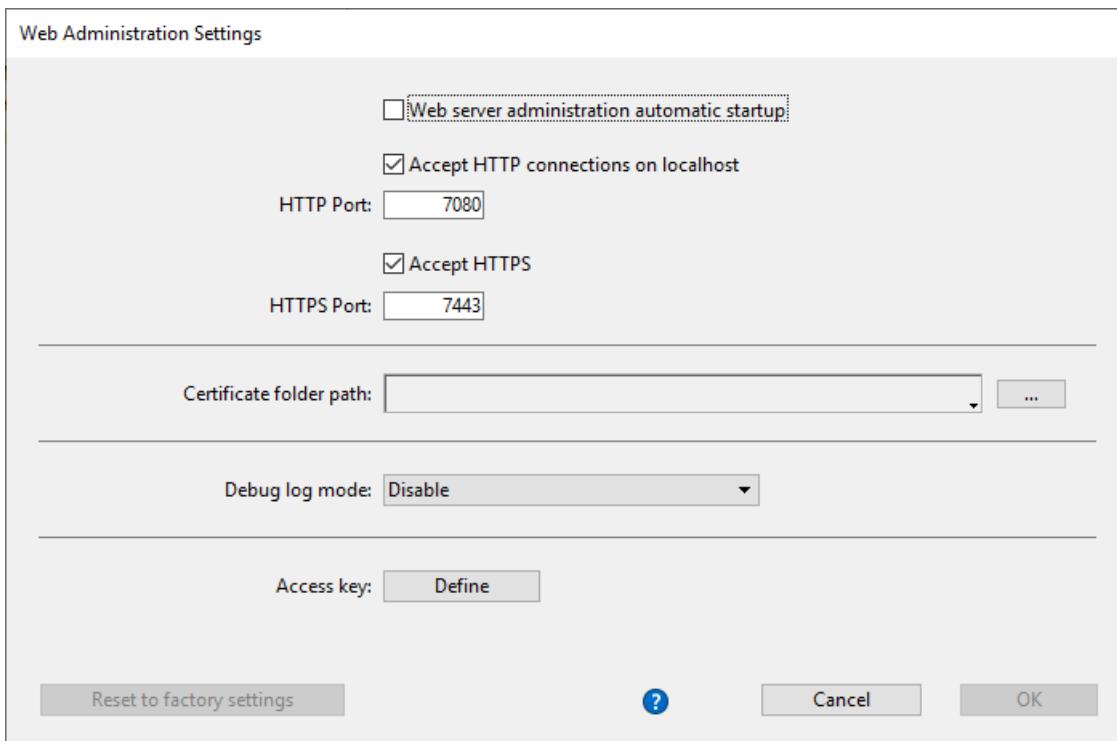
Si utiliza una aplicación 4D sin interfaz, puede utilizar los [argumentos de la interfaz de línea de comandos](#) para definir la configuración básica. Tendrá que personalizar el archivo de configuración para definir los parámetros avanzados.

## Caja de diálogo de parámetros

Para abrir la caja de diálogo de configuración de administración web, seleccione Archivo > Administración Web > Configuración....



Se muestra la siguiente caja de diálogo:



## Inicio automático de la administración del servidor web

Marque esta opción para lanzar el servidor web `WebAdmin` automáticamente cuando se inicie la aplicación 4D o 4D Server (ver [arriba](#)). Por defecto, esta opción no está seleccionada.

## Conexiones HTTP en localhost aceptadas

Cuando esta opción está seleccionada, podrá conectarse al servidor web `WebAdmin` a través de HTTP en la misma máquina que la aplicación 4D. Por defecto, esta opción está seleccionada.

### Notas:

- Nunca se aceptan conexiones con HTTP que no sean localhost.
- Incluso si esta opción está activada, cuando [HTTPS aceptada](#) está activada y la configuración TLS es válida, las conexiones localhost utilizan HTTPS.

## Puerto HTTP

Número de puerto a utilizar para las conexiones al servidor web `WebAdmin` a través de HTTP cuando la opción [Conexiones HTTP en localhost aceptadas](#) está marcada. El valor por defecto es 7080.

## Aceptar HTTPS

Cuando esta opción está seleccionada, podrá conectarse al servidor web `WebAdmin` a través de HTTPS. Por defecto, esta opción está seleccionada.

## Puerto HTTPS

Número de puerto a utilizar para las conexiones al servidor web `WebAdmin` a través de HTTPS cuando la opción HTTPS aceptada está marcada. El valor por defecto es 7443.

## Ruta de la carpeta de certificados

Ruta de la carpeta donde se encuentran los archivos del certificado TLS. Por defecto, la ruta de la carpeta de certificados está vacía y 4D o 4D Server utilizan los archivos de certificados contenidos en la aplicación 4D (los certificados personalizados deben almacenarse junto a la carpeta de proyecto).

## Modo de registro de depuración

Estado o formato del archivo de registro de peticiones HTTP (`HTTPDebugLog_nn.txt`, almacenado en la carpeta "Logs" de la aplicación -- *nn* es el número de archivo). Las siguientes opciones están disponibles:

- Desactivado (por defecto)
- Con todas las partes del cuerpo - habilitado con partes del cuerpo de las peticiones y respuestas
- Sin las partes del cuerpo - activado sin partes del cuerpo (se indica el tamaño del cuerpo)
- Con los cuerpos de las peticiones - activado con las partes del cuerpo únicamente en las peticiones
- Con la respuesta cuerpos - activado con las partes del cuerpo únicamente en las respuestas

## Llave de acceso

La configuración de una llave de acceso es obligatoria para desbloquear el acceso al servidor web `WebAdmin` a través de una URL (el acceso a través de un comando del menú 4D no requiere una llave de acceso). Cuando no se define ninguna llave de acceso, no se permite que ningún cliente web se conecte a través de una URL a una interfaz de administración web como la página [Explorador de datos](#). Se devuelve una página de error en caso de solicitud de conexión:



Una llave de acceso es similar a una contraseña pero no está asociada a un inicio de sesión.

- Para definir una nueva llave de acceso: haga clic en el botón `Definir`, introduzca la cadena de la llave de acceso en la caja de diálogo y haga clic en `OK`. La etiqueta del botón se convierte en `Modificar`.
- Para modificar la llave de acceso: haga clic en el botón `Modificar`, introduzca la nueva cadena de la llave de acceso en la caja de diálogo y haga clic en `OK`.
- Para eliminar la llave de acceso: haga clic en el botón `Modificar`, deje vacía el área de la llave de acceso y haga clic en `OK`.

## Configuración de WebAdmin sin interfaz

Todos [los parámetros de WebAdmin](#) se almacenan en el archivo `WebAdmin.4DSettings`. Hay un archivo `WebAdmin.4DSettings` por defecto para cada aplicación 4D y 4D Server, por lo que es posible desplegar varias aplicaciones en la misma máquina local.

Cuando se ejecuta una aplicación 4D o 4D Server sin interfaz, se puede configurar y utilizar el archivo `WebAdmin.4DSettings` por defecto, o designar un archivo `.4DSettings` personalizado.

Para establecer el contenido del archivo, puede utilizar la [ventana de parámetros WebAdmin](#) de la aplicación 4D con una interfaz y ejecutarla luego sin interfaz. Se utiliza entonces el archivo por defecto `WebAdmin.4DSettings`.

O bien, puede definir un archivo personalizado `.4DSettings` (formato xml) y utilizarlo en lugar del archivo

predeterminado. En la [Interfaz de línea de comandos](#) hay varios argumentos dedicados para soportar esta funcionalidad.

La clave de acceso no se almacena de manera transparente en el archivo `.4DSettings`.

Ejemplo:

```
"%HOMEPATH%\Desktop\4D Server.exe" MyApp.4DLink --webadmin-access-key  
"my Fabulous AccessKey" --webadmin-auto-start true  
--webadmin-store-settings
```

## Autenticación y sesión

- Cuando se accede a una página de gestión web introduciendo una URL y sin identificación previa, se requiere una autenticación. El usuario debe introducir la [Llave-de-acceso](#) en una ventana de autenticación. Si la Llave de acceso no fue definida en la configuración de `WebAdmin`, no es posible el acceso vía URL.
- Cuando se accede a una página de gestión web directamente desde un elemento de menú de 4D o 4D Server (como [Registros > Explorador de datos](#) o [Ventana > Explorador de datos \(4D Server\)](#)), el acceso se concede sin autenticación, el usuario se autentifica automáticamente.

Una vez concedido el acceso, se crea una [sesión web](#) con el privilegio "WebAdmin" en la aplicación 4D. Mientras la sesión actual tenga el privilegio "WebAdmin", el componente `WebAdmin` entrega las páginas solicitadas.

# Explorador de datos Web

Vista previa: El explorador de datos web se ofrece como una funcionalidad de visión general. No se recomienda utilizar esta funcionalidad en producción. La implementación final podría ser ligeramente diferente.

El Explorador de datos ofrece una interfaz web para ver y consultar los datos del almacén de datos de su proyecto. Utilizando esta herramienta, puede navegar fácilmente entre todas sus entidades y buscar, ordenar o filtrar los valores de los atributos. Le ayuda a controlar los datos y a identificar rápidamente los problemas en cualquier etapa del proceso de desarrollo.

ID	firstname	lastname	salary	birthdate
15	Hermance	ELNOJRAS	37 809	19/10/1996
26	Hermelin	SAGELAN	62 409	25/04/1975
41	Helixane	RAPPART	30 709	04/04/1971
65	Harrison	CHESSIT	78 109	17/07/1979
84	Hatrice	MERHEDIUR	62 109	06/04/1996
121	Heleazar	DROENT	74 209	02/10/1970
125	Harn	CIURSOAR	52 209	20/11/1985
139	Hectoria	BANIOT	33 809	09/07/1980
200	Harmonie	CESILO	72 509	11/02/1995
258	Harkaitz	MARZIUG	57 809	13/01/1980
271	Haiza	FEUCHA	59 509	10/03/1970
297	Helma	GARBAL	73 709	03/03/1975
372	Hélïena	ERASTA	45 709	15/01/1976

## Configuración

El Explorador de datos se basa en el componente servidor web [WebAdmin](#) para la configuración y los parámetros de autenticación.

- configuración: la configuración del Explorador de datos reutiliza los [parámetros del servidor web WebAdmin](#),
- autenticación: el acceso al Explorador de datos se concede cuando [el usuario de la sesión está autenticado](#) y tiene el privilegio "WebAdmin". Cuando se accede al Explorador de datos a través del elemento de menú Explorador de datos (ver más adelante), se proporciona una autenticación automática.

El acceso al Explorador de Datos puede desactivarse mediante la función `.setAdminProtection()`.

## Apertura del Explorador de datos

La página del Explorador de datos está disponible automáticamente cuando [el servidor web WebAdmin se inicia](#).

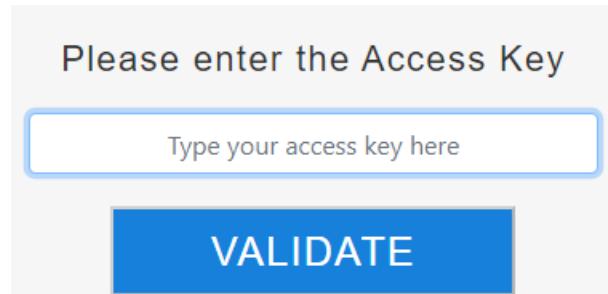
Para conectarse a la página web del Explorador de datos:

- si utiliza una aplicación 4D con interfaz, seleccione el comando Explorador de datos... de:

- o el menú Registros (en 4D monopuesto)
- o el menú Ventana (en 4D Server)
- tanto si utiliza una aplicación 4D sin interfaz como si no, puede abrir su navegador web e introducir la siguiente dirección:

`IPaddress:HTTPPort/dataexplorer or IPaddress:HTTPSPort/dataexplorer`

En este contexto, se le pedirá que introduzca la [clave de acceso](#) para abrir una sesión de `WebAdmin` en el servidor:



Los valores `HTTPPort` y `HTTPSPort` se configuran en los parámetros de `WebAdmin`.

## Uso del explorador de datos

Además de una vista completa y personalizable de sus datos, el Explorador de datos le permite consultar y ordenar sus datos.

### Requisitos

El Explorador de datos es compatible con los siguientes navegadores web:

- Chrome
- Safari
- Edge
- FireFox

La resolución mínima para utilizar el Explorador de Datos es de 1280x720. La resolución recomendada es de 1920x1080.

### Básicos

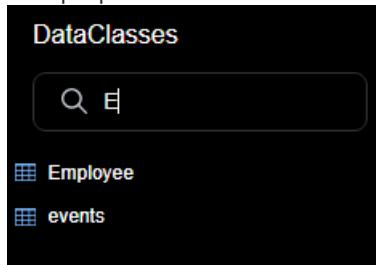
El Explorador de Datos ofrece un acceso global al modelo de datos ORDA con respecto a las reglas de mapeo [ORDA](#).

Puede pasar al tema de visualización modo oscuro utilizando el selector en la parte inferior de la página:

ID	Firstname	Lastname	salary	birthday
18	Hernane	ELNORIAS	37.809	18/10/1996
26	Hernane	SASIELAN	42.409	25/04/1975
41	Hernane	AARHATT	39.709	04/04/1971
65	Harrison	CHESSET	78.109	15/03/1979
84	Holot	MURIEGUR	62.109	08/04/1996
121	Helescar	GRENENT	74.209	02/10/1970
125	Horn	COURSIER	52.209	20/11/1985
139	Herofilia	BANOT	33.809	09/07/1980
200	Homomita	CESILIO	72.509	11/02/1995
258	Hofzahl	MARQUES	57.809	13/01/1980
271	Hoxia	PEUCHA	99.509	10/03/1970
287	Hoxia	GABRIAL	79.709	03/03/1975
372	Holma	GRATTA	45.709	18/01/1976

La página contiene varias áreas:

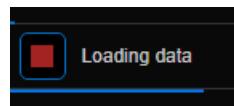
- En el lado izquierdo se encuentran el área de las Clases de datos y el área de los atributos, lo que permite seleccionar las clases de datos y los atributos a mostrar. Los atributos se ordenan según el orden de creación de la estructura subyacente. Las llaves primarias y los atributos indexados tienen un ícono específico. Puede filtrar la lista de nombres de clases de datos y de atributos propuestos utilizando las áreas de búsqueda respectivas.



- La parte central contiene el Área de búsqueda y la Rejilla de datos (lista de entidades de la clase de datos seleccionada). Cada columna de la cuadrícula representa un atributo del almacén de datos.
  - Por defecto, se muestran todas las entidades. Puede filtrar las entidades mostradas utilizando el área de búsqueda. Hay dos modos de consulta disponibles: [Consulta sobre atributos](#) (seleccionada por defecto), y la [Consulta avanzada con expresión](#). El modo de consulta se selecciona haciendo clic en el botón correspondiente (el botón X permite restablecer el área de consulta y, por tanto, dejar de filtrar):



- El nombre de la clase de datos seleccionada se añade como una pestaña encima de la cuadrícula de datos. Utilizando estas pestañas, puede cambiar entre las clases de datos que ya han sido seleccionadas. Puede eliminar una clase de datos referenciada haciendo clic en el ícono "eliminar" situado a la derecha del nombre de la clase de datos.
- Puede reducir el número de columnas desmarcando los atributos en la parte izquierda. También puede cambiar las columnas de la cuadrícula de datos utilizando arrastrar y soltar. Puede hacer clic en el encabezado de una columna para [ordenar entidades](#) de acuerdo a sus valores (cuando sea posible).
- Si una operación requiere mucho tiempo, se muestra una barra de progreso. Puede detener la operación en curso en cualquier momento haciendo clic en el botón rojo:



- En el lado derecho está el área de Detalles: muestra los valores de los atributos de la entidad seleccionada actualmente. Se muestran todos los tipos de atributos, incluidas las imágenes y los objetos (expresados en json). Puede navegar entre las entidades de la clase de datos haciendo clic en los enlaces Primero / Anterior / Siguiente / Último en la parte inferior del área.

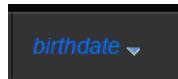
## Actualizar los contenidos

Cuando el modelo ORDA o los datos se modifican del lado de la base de datos (se añade una tabla, se edita o se elimina un registro, etc.), basta con actualizar la página del Explorador de datos en el navegador (utilizando la tecla F5, por ejemplo).

## Ordenar las entidades

Puede reordenar la lista de entidades mostrada según los valores de los atributos. Todos los tipos de atributos pueden utilizarse para una ordenación, excepto la imagen y el objeto.

- Haga clic en el encabezado de una columna para ordenar entidades de acuerdo a los valores de atributo correspondientes. Por defecto, la ordenación es ascendente. Haga clic dos veces para una ordenación descendente. Una columna utilizada para ordenar las entidades se muestra con un pequeño ícono y su nombre está en *ítälica*.



- Puede ordenar los atributos en varios niveles. Por ejemplo, puede ordenar a los empleados por ciudad y luego por salario. Para ello, mantenga presionada la tecla Mayús y haga clic sucesivamente en el encabezado de cada

columna a incluir en el orden de clasificación.

## Consultas basadas en atributos

En este modo, puede filtrar las entidades introduciendo los valores que desea encontrar (o excluir) en las áreas situadas arriba de la lista de atributos. Puede filtrar por uno o varios atributos. La lista de entidades se actualiza automáticamente cuando se digita.

ID	firstname
1	Florette
29	Flora
1 200	Floria

Si introduce varios atributos, se aplica automáticamente un AND. Por ejemplo, el siguiente filtro muestra las entidades con el atributo `firstname` que empieza por "flo" AND el valor del atributo `salary > 50000`:

firstname	salary
Flora	76 109
Floria	58 009
Florent	79 409

El botón X permite eliminar los atributos introducidos y así dejar de filtrar.

Existen diferentes operadores y opciones de consulta, según el tipo de datos del atributo.

No se puede filtrar por atributos de imagen o de objeto.

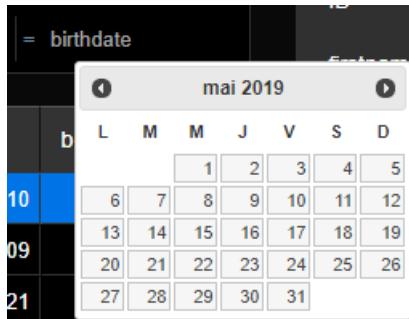
### Operadores numéricos

Con los atributos numéricos, de fecha y de hora, el operador "=" está seleccionado por defecto. Sin embargo, puede seleccionar otro operador de la lista de operadores (haga clic en el icono "=" para mostrar la lista):



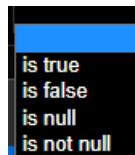
### Fechas

Con los atributos de fecha, puede introducir la fecha a utilizar a través de un widget de selección de fecha (haga clic en el área de la fecha para mostrar el calendario):



## Booleanos

Al hacer clic en un área de atributos booleanos, se puede filtrar sobre los valores true/false pero también los valores null/not null:



- null indica que el valor del atributo no fue definido
- no nulo indica que el valor del atributo está definido (por tanto, true o false).

## Texto

Los filtros texto no son diacríticos (a = A).

El filtro es del tipo "empieza por". Por ejemplo, al introducir "Jim" se mostrarán los valores "Jim" y "Jimmy".

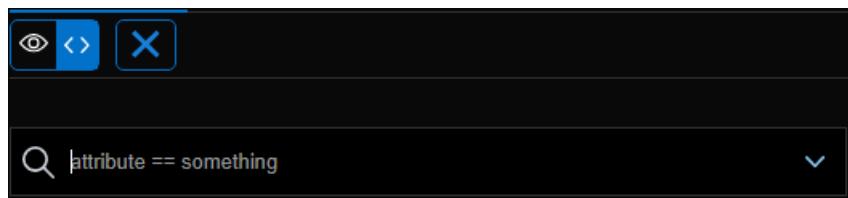
También puede utilizar el carácter comodín (@) para sustituir uno o varios caracteres iniciales. Por ejemplo:

Un filtro con	Resultados
Bel	Todos los valores que empiezan por "Bel"
@do	Todos los valores que contienen "do"
Bel@do	Todos los valores que empiezan por "Bel" y contienen "do"

Si desea crear consultas más específicas, como "es exactamente", es posible que tenga que utilizar la función de consultas avanzadas.

## Consultas avanzadas con expresión

Cuando se selecciona esta opción, aparece un área de consulta sobre la lista de entidades, que permite introducir cualquier expresión para filtrar el contenido:



Puede introducir consultas avanzadas que no están disponibles como consultas de atributos. Por ejemplo, si quiere encontrar entidades con el atributo `firstname` que contenga "Jim" pero no "Jimmy", puede escribir:

```
firstname=="Jim"
```

Puede utilizar cualquier expresión de consulta ORDA como [documentada con la función query\(\)](#), con las siguientes limitaciones o diferencias:

- Por seguridad, no se pueden ejecutar fórmulas utilizando `eval()`.

- No se pueden utilizar marcadores de posición; hay que escribir un `queryString` con valores.
- Los valores de las cadenas que contienen caracteres de espacio deben ir entre comillas dobles ("").

Por ejemplo, con la clase de datos Employee, puede escribir:

```
firstname = "Marie Sophie" AND manager.lastname = "@th"
```

Puede hacer clic en el icono para mostrar tanto `queryPlan` como `queryPath`. En el área, puede pasar sobre los bloques de subconsultas para tener información detallada por subconsulta:



Haga clic derecho en el área de consulta para mostrar las anteriores consultas válidas:

ID	firstname	lastname	salary
2	isabelle	Maes	30000
3	Narita	Yamaguchi	30000

A context menu is open over the third row, listing several valid query strings:

- firstname = "h@" AND lastname == "sm@" OR salary > 30000
- firstname = "h@" AND lastname == "sm@"
- firstname = "h@"
- firstname == "Jim"
- firstname = "M" and lastname = "@th"
- firstname = "M@" and lastname = "@th"
- firstname = "M@" or lastname = "@th"
- firstname = "h@" AND lastname == "smith" OR salary > 30000

# Interfaz de línea de comando

Puede utilizar el terminal macOS o la consola Windows para manejar sus aplicaciones 4D (4D y 4D Server) utilizando líneas de comando. Más concretamente, esta funcionalidad le permite:

- lanzar una base de datos de forma remota, lo que puede ser especialmente útil para administrar los servidores web.
- ejecutar pruebas automáticas para sus aplicaciones.

## Información básica

Puede ejecutar líneas de comando para las aplicaciones 4D utilizando el terminal macOS o la consola Windows.

- En macOS, debe utilizar el comando `open`.
- En Windows, puede pasar los argumentos directamente.

En macOS, se pueden pasar los argumentos directamente yendo a la carpeta donde se encuentra la aplicación dentro del paquete (Contents/MacOS), lo que permite direccionar el flujo stderr. Por ejemplo, si el paquete 4D se encuentra en la carpeta `MyFolder`, debe escribir la línea de comandos de la siguiente manera:

`/MyFolder/4D.app/Contents/MacOS/4D`. Sin embargo, le recomendamos que utilice el comando `open` siempre que no necesite acceder al flujo stderr.

## Lanzar una aplicación 4D

A continuación se describen las líneas de comando y los argumentos soportados para lanzar aplicaciones 4D.

Sintaxis:

```
<applicationPath> [--version] [--help] [--project] [<projectPath | packagePath | 4dlinkPath> [--data <dataPath>] [--opening-mode interpreted | compiled] [--create-data] [--user-param <user string>] [--headless] [--display <display>] [--webadmin-settings-file] [--webadmin-access-key] [--webadmin-auto-start] [--webadmin-store-settings]
```

Argumento	Valor	Descripción
<code>applicationPath</code>	Ruta de acceso a 4D, 4D Server o de la aplicación fusionada	Lance la aplicación. Es idéntico a hacer doble clic en la aplicación 4D. Cuando se llama sin argumento de archivo de estructura, la aplicación se ejecuta y aparece la caja de diálogo "seleccionar la base de datos".
<code>--version</code>		Muestra la versión de la aplicación y sale
<code>--help</code>		Muestra el mensaje de ayuda y salir. Argumentos alternativos: <code>-?</code> , <code>-h</code>
<code>--project</code>	<code>projectPath   packagePath   4dlinkPath</code>	Archivo de proyecto a abrir con el archivo de datos actual. No aparece ninguna caja de diálogo.
<code>--data</code>	<code>dataPath</code>	Archivo de datos a abrir con el archivo de proyecto designado. Si no se especifica, 4D utiliza el último archivo de datos abierto.
<code>--opening-mode</code>	<code>interpreted   compiled</code>	Base de datos de peticiones a abrir en modo interpretado o compilado. No se lanza ningún error si el modo solicitado no está disponible.

Argumento	Valor	Descripción
--data		Creación automática de un nuevo archivo de datos si no se encuentra un archivo de datos válido. No aparece ninguna caja de diálogo. 4D utiliza el nombre del archivo pasado en el argumento "--data" si lo hay (genera un error si ya existe un archivo con el mismo nombre).
--user-param	Cadena usuario personalizada	Una cadena que estará disponible en la aplicación 4D a través del comando Get database parameter (la cadena no debe comenzar por un carácter "-", que está reservado).
--headless		<p>Lanza 4D, 4D Server o la aplicación fusionada sin interfaz (modo headless). En este modo:</p> <ul style="list-style-type: none"> <li>• El modo Diseño no está disponible, la base de datos se inicia en modo Aplicación</li> <li>• No se muestra ninguna barra de herramientas, barra de menús, ventana MDI o pantalla de inicio</li> <li>• No aparece ningún ícono en el dock o en la barra de tareas</li> <li>• La base de datos abierta no está registrada en el menú "Bases de datos recientes"</li> <li>• El registro de diagnóstico se lanza automáticamente ( ver <a href="#">SET DATABASE PARAMETER</a>, selector 79)</li> <li>• Cada llamada a un cuadro de diálogo es interceptada y se proporciona una respuesta automática (por ejemplo, OK para el comando <a href="#">ALERT</a>, Abort para un diálogo de error...). Todos los comandos interceptados(*) se registran en el registro de diagnóstico.</li> </ul> <p>Para las necesidades de mantenimiento, puede enviar cualquier texto a los flujos de salida estándar utilizando la función <a href="#">LOG EVENT</a>. Tenga en cuenta que las aplicaciones 4D sin interfaz sólo pueden cerrarse mediante una llamada a <a href="#">QUIT 4D</a> utilizando el administrador de tareas del sistema operativo.</p>
--dataless		<p>Lanza 4D, 4D Server o la aplicación fusionada en modo sin datos. El modo sin datos es útil cuando 4D ejecuta tareas sin necesidad de datos (compilación de proyectos, por ejemplo).</p> <p>En este modo:</p> <ul style="list-style-type: none"> <li>• No se abre ningún archivo que contenga datos, incluso si se especifica en la línea de comandos o en el archivo <a href="#">.4DLink</a>, o cuando se utilizan los comandos <a href="#">CREATE DATA FILE</a> y <a href="#">OPEN DATA FILE</a>.</li> <li>• Los comandos que manipulan datos generarán un error. Por ejemplo, <a href="#">CREATE RECORD</a> lanza "no hay tabla a la que aplicar el comando".</li> </ul> <p>Nota:</p> <ul style="list-style-type: none"> <li>• Si se pasa en la línea de comandos, el modo sin datos se aplica a todas las bases de datos abiertas en 4D, mientras no se cierre la aplicación.</li> <li>• Si se pasa utilizando el archivo <a href="#">.4DLink</a>, el modo sin datos sólo se aplica a la base de datos especificada en el archivo <a href="#">.4DLink</a>. Para más información sobre los archivos <a href="#">.4DLink</a>, ver <a href="#">Atajos para abrir proyectos</a>.</li> </ul>
--webadmin-settings-file	Ruta del archivo	Ruta del archivo WebAdmin <a href="#">.4DSettings</a> personalizado para el <a href="#">servidor web WebAdmin</a>
--webadmin-access-key	Cadena	Llave de acceso al <a href="#">servidor web WebAdmin</a>
--webadmin-auto-start	Booleano	Estado del inicio automático del <a href="#">servidor web WebAdmin</a>

Argumento	Valor	Descripción
<code>--webadmin-store-settings</code>		Almacena la llave de acceso y los parámetros de inicio automático en el archivo de parámetros actual (es decir, el archivo <code>WebAdmin.4DSettings</code> por defecto o un archivo personalizado designado por el parámetro <code>--webadmin-settings-path</code> ). Utilice el argumento <code>--webadmin-store-settings</code> para guardar estos parámetros si es necesario

**Diagnostic log file** (licence alert, conversion dialog, database selection, data file selection). En este caso, se lanza un mensaje de error tanto en el flujo stderr como en el registro de eventos sistema, y luego la aplicación se cierra.

## Ejemplos

Estos ejemplos suponen que su aplicación 4D está almacenada en el escritorio y que la base de datos a abrir se encuentra en la carpeta "Documentos".

La carpeta actual del usuario se alcanza utilizando el comando "`~`" en macOS y el comando "`%HOMEPATH%`" en Windows.

Lance la aplicación:

- macOS:

```
open ~/Desktop/4D.app
```

- Windows:

```
%HOMEPATH%\Desktop\4D\4D.exe
```

Lanzar la aplicación con un archivo paquete en macOS:

```
yarn open ~/Desktop/4D.app --args ~/Documents/myDB.4dbase
```

Lanzar la aplicación con un archivo proyecto:

- macOS:

```
yarn open ~/Desktop/4D.app --args ~/Documents/myProj/Project/myProj.4DProject
```

- Windows:

```
%HOMEPATH%\Desktop\4D\4D.exe %HOMEPATH%\Documents\myProj\Project\myProj.4DProject
```

Lanzar la aplicación con un archivo proyecto y un archivo de datos:

- macOS:

```
open ~/Desktop/4D.app --args --project ~/Documents/myProj/Project/myProj.4DProject --data ~/Documents/da
```

- Windows:

```
%HOMEPATH%\Desktop\4D\4D.exe --project %HOMEPATH%\Documents\myProj\Project\myProj.4DProject --data %HOME  
0:  
%HOMEPATH%\Desktop\4D\4D.exe /project %HOMEPATH%\Documents\myProj\Project\myProj.4DProject /data %HOMEP
```

Lanzar la aplicación con un archivo .4DLink:

- macOS:

```
open ~/Desktop/4D.app MyDatabase.4DLink
```

```
open "~/Desktop/4D Server.app" MyDatabase.4DLink
```

- Windows:

```
%HOMEPATH%\Desktop\4D.exe MyDatabase.4DLink
```

```
%HOMEPATH%\Desktop\4D Server.exe" MyDatabase.4DLink
```

Lanzar la aplicación en modo compilado y crear un archivo de datos si no está disponible:

- macOS:

```
open ~/Desktop/4D.app ~/Documents/myBase.4dbase --args --opening-mode compiled --create-data true
```

- Windows:

```
%HOMEPATH%\Desktop\4D\4D.exe %HOMEPATH%\Documents\myBase.4dbase\myDB.4db --opening-mode compiled --creat
```

Lanzar la aplicación con un archivo proyecto y un archivo de datos y pasar una cadena como parámetro usuario:

- macOS:

```
open ~/Desktop/4D.app --args --project ~/Documents/myProj/Project/myProj.4DProject --data ~/Documents/da
```

- Windows:

```
%HOMEPATH%\Desktop\4D\4D.exe --project %HOMEPATH%\Documents\myProj\Project\myProj.4DProject --data %HOME
```

Lanzar la aplicación sin interfaz (modo headless):

- macOS:

```
open ~/Desktop/4D.app --args --project ~/Documents/myProj/Project/myProj.4DProject --data ~/Documents/da
```

```
open ~/Desktop/MyBuiltRemoteApp --headless
```

- Windows:

```
%HOMEPATH%\Desktop\4D\4D.exe --project %HOMEPATH%\Documents\myProj\Project\myProj.4DProject --data %HOME%  
%HOMEPATH%\Desktop\4D\MyBuiltRemoteApp.exe --headless
```

# Protocolo TLS (HTTPS)

Todos los servidores 4D pueden comunicarse en modo seguro a través del protocolo TLS (Transport Layer Security):

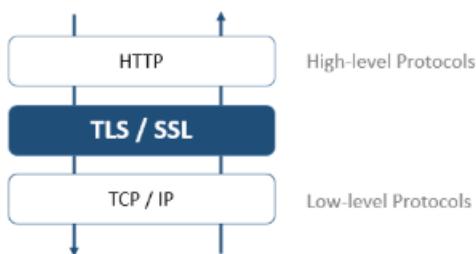
- el servidor web
- el servidor de aplicaciones (aplicaciones de escritorio cliente-servidor)
- el servidor SQL

## Generalidades

El protocolo TLS (sucesor de SSL) ha sido diseñado para asegurar los intercambios de datos entre dos aplicaciones, principalmente entre un servidor web y un navegador. Este protocolo es ampliamente utilizado y es compatible con la mayoría de los navegadores web.

A nivel de red, el protocolo de seguridad se inserta entre la capa TCP/IP (nivel bajo) y el protocolo de alto nivel HTTP. Ha sido diseñado principalmente para trabajar con HTTP.

Configuración de la red utilizando TSL:



El protocolo TLS está diseñado para autenticar al emisor y al receptor y para garantizar la confidencialidad e integridad de la información intercambiada:

- Autenticación: se confirma la identidad del emisor y del receptor.
- Confidencialidad: los datos enviados se cifran para que ninguna tercera persona pueda entender el mensaje.
- Integridad: los datos recibidos no han sido modificados, por accidente o de forma malintencionada.

TLS utiliza una técnica de cifrado de llave pública basada en un par de llaves asimétricas para el cifrado y el descifrado: una llave pública y una llave privada. La llave privada se utiliza para encriptar los datos. El remitente (el sitio web) no se la da a nadie. La llave pública se utiliza para descifrar la información y se envía a los receptores (navegadores web) a través de un certificado. Cuando se utiliza TLS con Internet, el certificado se entrega a través de una autoridad de certificación, como Verisign®. El sitio web paga a la Autoridad de Certificación para que le entregue un certificado que garantice la autenticación del servidor y que contenga la llave pública que permite el intercambio de datos de forma segura.

Para más información sobre el método de encriptación y los temas de llave pública y privada, consulte la descripción del comando `ENCRYPT BLOB`.

## Versión mínima

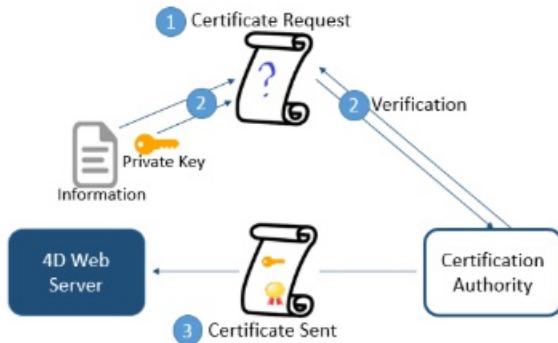
Por defecto, la versión mínima del protocolo seguro que acepta el servidor es TLS 1.2. Puede modificar este valor utilizando `Min TLS version` con el comando `SET DATABASE PARAMETER`.

Puede controlar el nivel de seguridad de su servidor web definiendo la [versión TLS mínima](#) aceptada para las conexiones.

## ¿Cómo obtener un certificado?

Un servidor que funciona en modo seguro significa que necesita un certificado digital de una autoridad de certificación. Este certificado contiene diversa información, como el ID del sitio, así como la llave pública utilizada para comunicarse con el servidor. Este certificado se transmite a los clientes (por ejemplo, los navegadores web) que se conectan a este servidor. Una vez identificado y aceptado el certificado, la comunicación se realiza en modo seguro.

Los navegadores web sólo autorizan los certificados emitidos por una autoridad de certificación referenciada en sus propiedades.



La autoridad de certificación se elige en función de varios criterios. If the certification authority is well known, the certificate will be authorized by many browsers, however the price to pay will be expensive.

Para obtener un certificado digital:

1. Genere una llave privada utilizando el comando `GENERATE ENCRYPTION KEYPAIR`.

Atención: por razones de seguridad, la llave privada debe mantenerse siempre en secreto. En realidad, debería permanecer siempre en la máquina del servidor. Para el servidor web, el archivo Key.pem debe colocarse en la carpeta Project.

2. Utilice el comando `GENERATE CERTIFICATE REQUEST` para emitir una solicitud de certificado.

3. Envíe la solicitud de certificado a la autoridad de certificación elegida.

Para llenar una solicitud de certificado, es posible que tenga que ponerse en contacto con la autoridad de certificación. La autoridad de certificación comprueba que la información transmitida es correcta. La petición de certificado se genera en un BLOB utilizando el formato PKCS codificado en base64 (formato PEM). Este principio permite copiar y pegar las llaves como texto y enviarlas por correo electrónico sin modificar el contenido de la llave. Por ejemplo, puede guardar el BLOB que contiene la solicitud de certificado en un documento de texto (utilizando el comando `BLOB TO DOCUMENT`), y luego abrirlo y copiar y pegar su contenido en un correo o un formulario web para enviarlo a la autoridad de certificación.

4. Una vez que tenga el certificado, cree un archivo de texto llamado "cert.pem" y pegue en él el contenido del certificado.

Puede recibir un certificado de diferentes maneras (normalmente por correo electrónico o formulario HTML). 4D acepta todos los formatos de texto relacionados con la plataforma para los certificados (OS X, PC, Linux, etc.). Sin embargo, el certificado debe estar en formato PEM, \*es decir, \*, PKCS codificado en base64.

Los caracteres de fin de línea CR no se soportan por sí solos; debe utilizar CRLF o LF.

5. Coloque el archivo "cert.pem" en la [ubicación adecuada](#).

El servidor 4D puede ahora trabajar en modo seguro. El certificado tiene una validez de entre 3 meses y un año.

## Instalación y activación

Instalar los archivos `key.pem` y `cert.pem`

Para poder utilizar el protocolo TLS con el servidor, debe instalar el `key.pem` (documento que contiene la llave privada de cifrado) y `cert.pem` (documento que contiene el certificado) en la ubicación adecuada). Se requieren diferentes ubicaciones en función del servidor en el que se quiera utilizar TLS.

Los archivos `key.pem` y `cert.pem` por defecto se entregan con 4D. Para un mayor nivel de seguridad, le recomendamos encarecidamente que sustituya estos archivos por sus propios certificados.

## Con el servidor Web

Para ser utilizado por el servidor web de 4D, los archivos `key.pem` y `cert.pem` deben ser colocados:

- con 4D en modo local o 4D Server, junto a la [carpeta del proyecto](#)
- con 4D en modo remoto, en la carpeta de la base cliente en la máquina remota (para más información sobre la ubicación de esta carpeta, ver el comando [Get 4D folder](#) ).

Debe copiar estos archivos manualmente en la máquina remota.

## Con el servidor de aplicaciones (aplicaciones de escritorio cliente-servidor)

Para ser utilizado por el servidor de aplicaciones de 4D, los archivos `key.pem` y `cert.pem` deben ser colocados:

- en la carpeta [Resources](#) de la aplicación 4D Server
- y en la carpeta Resources de cada aplicación 4D remota (para más información sobre la ubicación de esta carpeta, ver el comando [Get 4D folder](#) ).

## Con el servidor Web

Para ser utilizado por el servidor SQL de 4D, los archivos `key.pem` y `cert.pem` deben ubicarse junto a la [carpeta del proyecto](#).

## Activar TLS

La instalación de los archivos `key.pem` y `cert.pem` permite utilizar TLS con el servidor 4D. Sin embargo, para que las conexiones TLS sean aceptadas por el servidor, debe habilitarlas:

- Con el servidor web 4D, debe [activar HTTPS](#). Puede definir la opción [HSTS](#) para redirigir a los navegadores que intenten conectarse en modo http.
- Con el servidor de aplicaciones, debe seleccionar la opción [Encriptar las comunicaciones cliente-servidor](#) en la página "Opciones cliente-servidor/red" del cuadro de diálogo Parámetros.
- Con el servidor SQL, debe seleccionar la opción [Habilitar TLS](#) en la página "SQL" del cuadro de diálogo Parámetros.

El servidor web 4D también soporta la opción [HSTS](#) para declarar que los navegadores sólo deben interactuar con él a través de conexiones seguras HTTPS.

## Perfect Forward Secrecy (PFS)

[PFS](#) añade una capa adicional de seguridad a sus comunicaciones. En lugar de utilizar llaves de intercambio predefinidas, PFS crea llaves de sesión de forma cooperativa entre las partes que se comunican utilizando algoritmos Diffie-Hellman (DH). La forma conjunta en que se construyen las llaves crea un "secreto compartido" que impide que partes externas puedan comprometerlas.

Cuando se habilita TLS en el servidor, PFS se habilita automáticamente. Si el archivo `dhpamrs.pem` (documento que contiene la llave privada DH del servidor) no existe todavía, 4D lo generará automáticamente con un tamaño de llave de 2048. La generación inicial de este archivo puede llevar varios minutos. El archivo se coloca con los archivos `key.pem` y `cert.pem`.

Si utiliza una [lista de cifrado personalizada](#) y desea habilitar el PFS, debe comprobar que contiene entradas con algoritmos DH o ECDH (Elliptic-curve Diffie-Hellman).



# Gestión de licencias 4D

Una vez instalados en su disco, debe activar sus productos 4D para poder utilizarlos. Normalmente, la activación es automática si [inicia sesión con su cuenta 4D](#) en el asistente de bienvenida.

Sin embargo, en algunos casos específicos podría ser necesario activar las licencias manualmente, por ejemplo si:

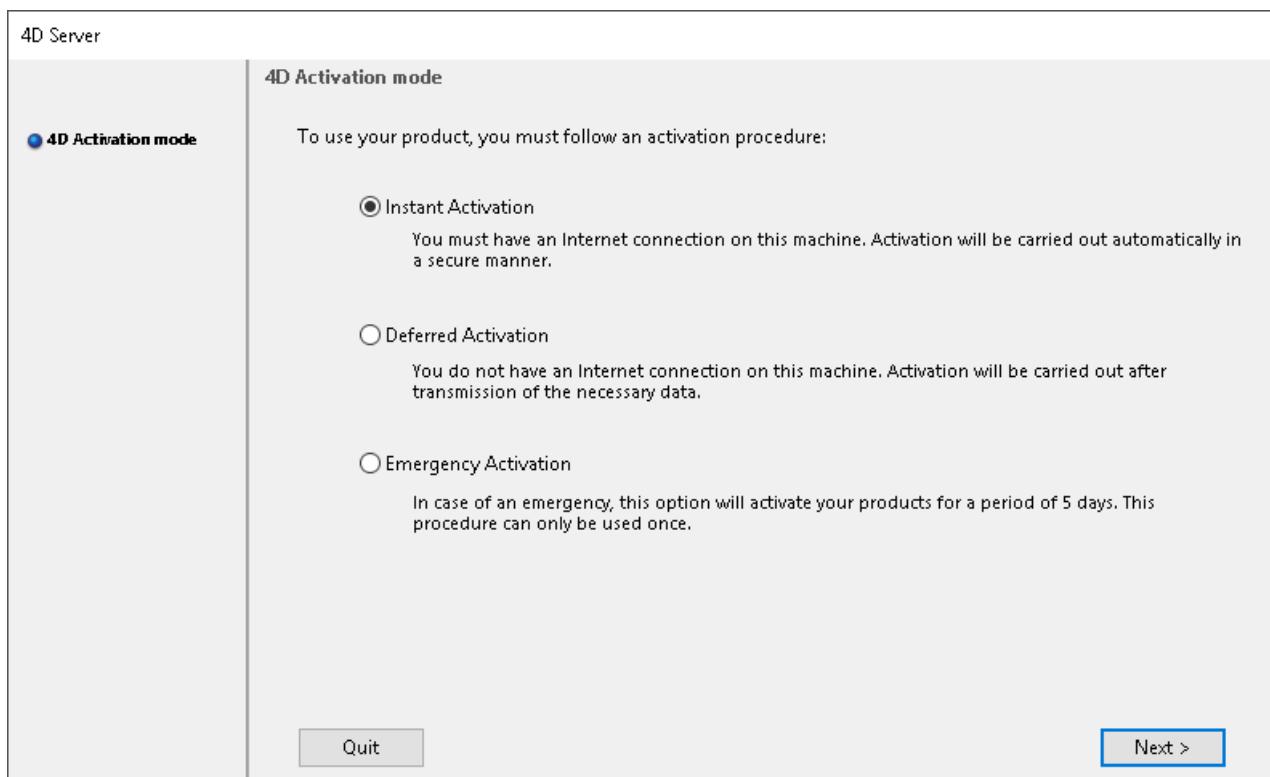
- su configuración no permite la activación automática,
- ha adquirido licencias adicionales.

No es necesaria la activación para los siguientes usos:

- 4D utilizado en modo remoto (conexión a un 4D Server)
- 4D utilizado en modo local con un proyecto aplicación interpretado sin acceso al entorno Diseño.

## Primera activación

Con 4D, seleccione el comando Gestión de licencias... del menú Ayuda. Con 4D Server, basta con lanzar la aplicación 4D Server. Aparece la caja de diálogo para elegir el [modo de activación](#).

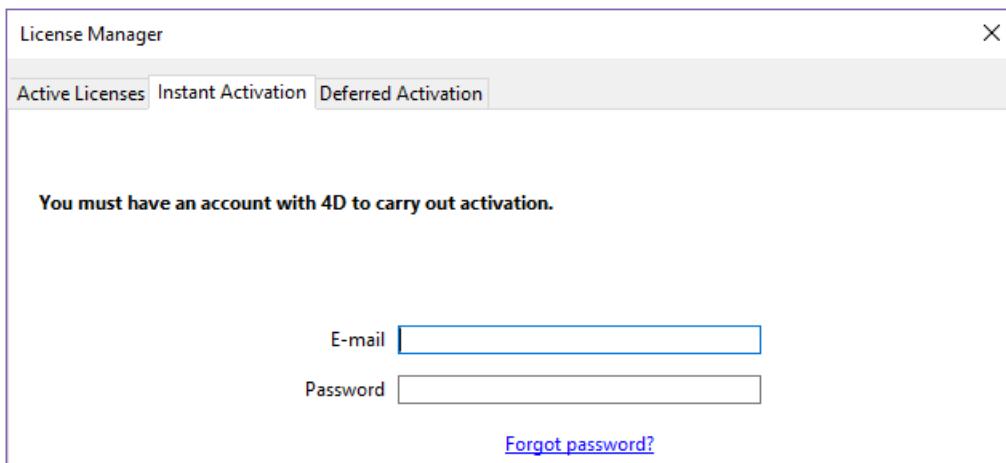


4D ofrece tres modos de activación. Recomendamos La activación inmediata.

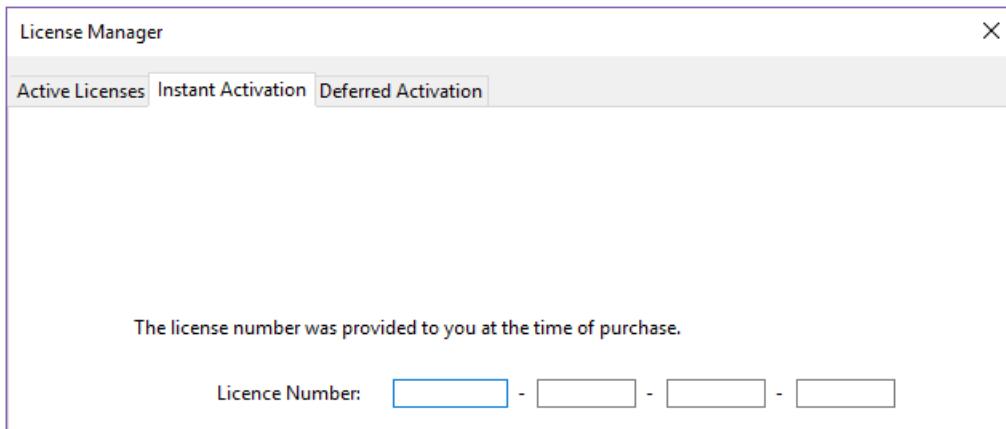
## Activación inmediata

Introduzca su identificación de usuario (correo electrónico o cuenta 4D) así como su contraseña. Si no tiene una cuenta de usuario, deberá crearla en la siguiente dirección:

<https://account.4d.com/us/login.shtml>



A continuación, introduzca el número de licencia del producto que desea activar. Este número se facilita por correo electrónico o por correo tras la compra de un producto.



## Activación diferida

Si no puede utilizar [la activación instantánea](#) porque su ordenador no tiene acceso a Internet, proceda a la activación diferida siguiendo los siguientes pasos.

1. En la ventana del Administrador de licencias, seleccione la pestaña Activación diferida.
2. Introduzca el número de licencia y su dirección de correo electrónico y, a continuación, haga clic en Generar el archivo para crear el archivo de identificación (*reg.txt*).

License Manager X

Active Licenses Instant Activation Deferred Activation

**Step 1 out of 3**

I want to generate an ID file that I will send to 4D in order to get an activation key in return.

→ Licence Number:  -  -  -

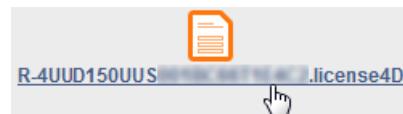
→ E-mail (mandatory):

→ Generate file...

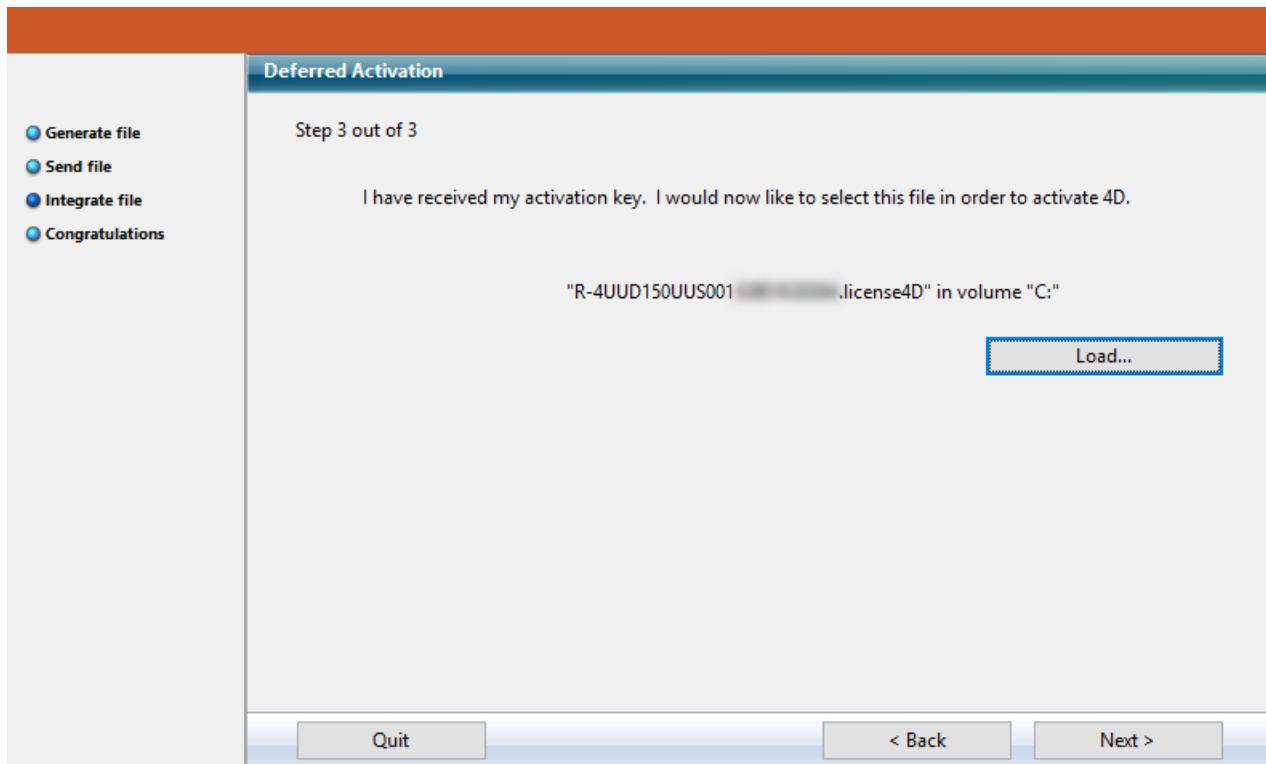
I have received my activation key. I would now like to select this key in order to activate 4D.

[< Back](#) [Next >](#) [Done](#)

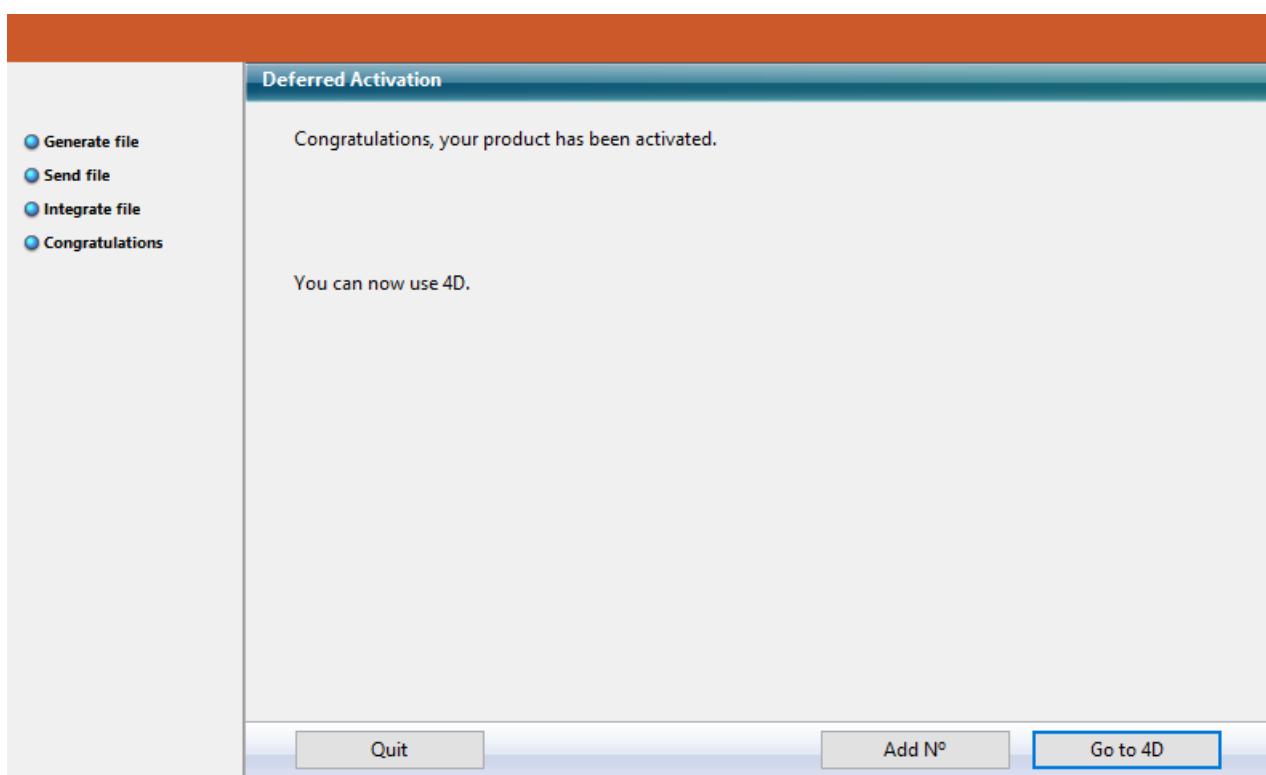
3. Guarde el archivo *reg.txt* en una unidad USB y llévelo a un ordenador que tenga acceso a Internet.
4. En la máquina con acceso a Internet, inicie sesión en <https://activation.4d.com>.
5. En la página Web, haga clic en el botón *Elegir archivo...* y seleccione el archivo *reg.txt* de los pasos 3 y 4; a continuación, haga clic en el botón *Activar*.
6. Descargue los archivos seriales.



7. Guarde el(s) archivo(s) *licencia4d* en un medio compartido y transfíralo(s) de nuevo a la máquina 4D del paso 1.
8. Ahora, de vuelta en la máquina con 4D, todavía en la página Activación Diferida, haga clic en *Siguiente*; a continuación, haga clic en el botón *Cargar...* y seleccione un archivo *licencia4d* del medio compartido del paso 7.



Con el archivo de licencia cargado, haga clic en Siguiente.



9. Haga clic en el botón Añadir Nº para añadir otra licencia. Repita estos pasos hasta que se hayan integrado todas las licencias del paso 6.

Su aplicación 4D está ahora activada.

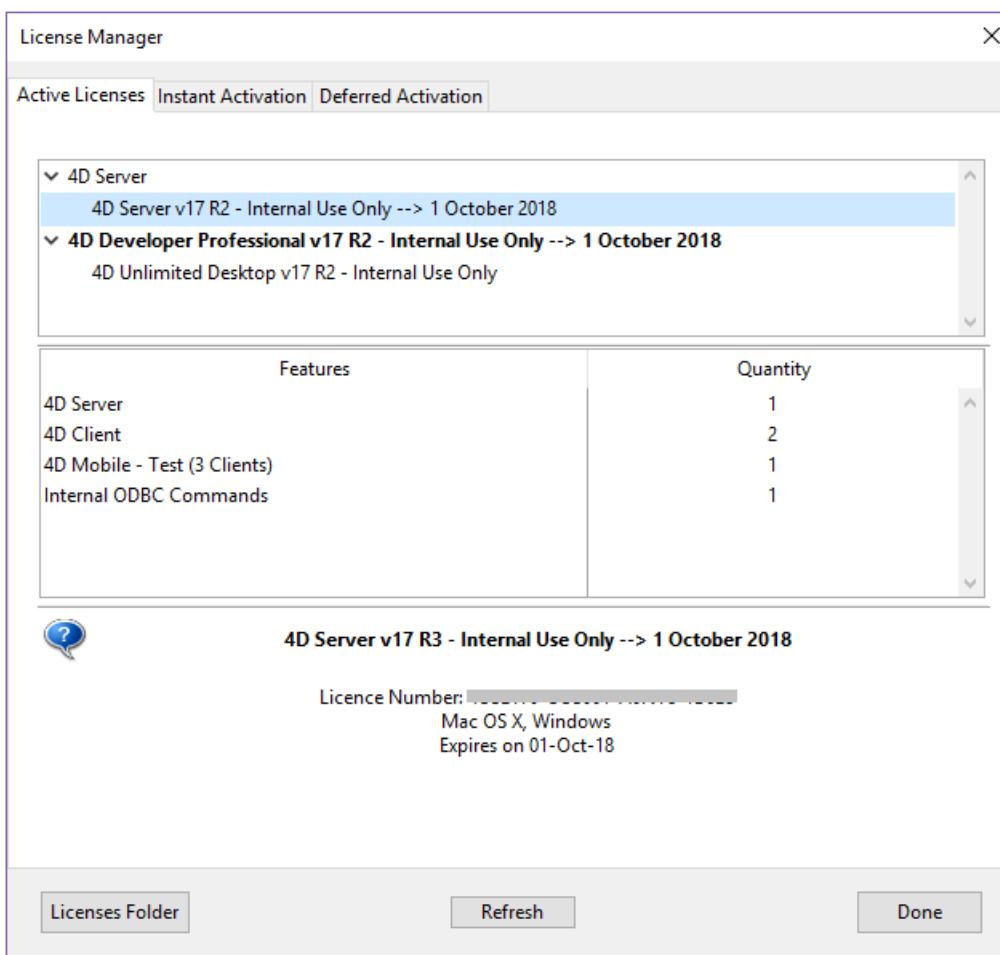
## Activación de emergencia

Este modo puede utilizarse para una activación temporal especial de 4D (5 días como máximo) sin conectarse al sitio web de 4D. Esta activación sólo puede utilizarse una vez.

## Añadir las licencias

Puede añadir nuevas licencias, por ejemplo para ampliar las capacidades de su aplicación, en cualquier momento.

Elija el comando Administrador de licencias... del menú Ayuda de la aplicación 4D o 4D Server, y luego haga clic en el botón Refrescar:



Este botón lo conecta con nuestra base clientes y activa automáticamente todas las licencias nueva o actualizadas relacionadas con la licencia actual (la licencia actual se muestra en negrita en la lista de "Licencias activas"). Sólo se le pedirá su cuenta de usuario y su contraseña.

- Si ha adquirido expansiones adicionales para un servidor 4D, no es necesario introducir ningún número de licencia, simplemente haga clic en Refrescar.
- En la primera activación de un 4D Server, basta con introducir el número de servidor y todas las expansiones adquiridas se asignan automáticamente.

Puede utilizar el botón Refrescar en los siguientes contextos:

- Cuando haya comprado una expansión adicional y quiera activarla,
- Cuando necesite actualizar un número temporal caducado (Partners o evoluciones).

## 4D Online Store

En 4D Store, puede pedir, actualizar, extender y/o gestionar los productos 4D. Puede llegar a la tienda en la siguiente dirección: (deberá seleccionar su país).

Haga clic en Inicio de sesión para acceder con su cuenta actual o en Nueva cuenta para crear una nueva, y siga las instrucciones que aparecen en pantalla.

## Gestión de licencias

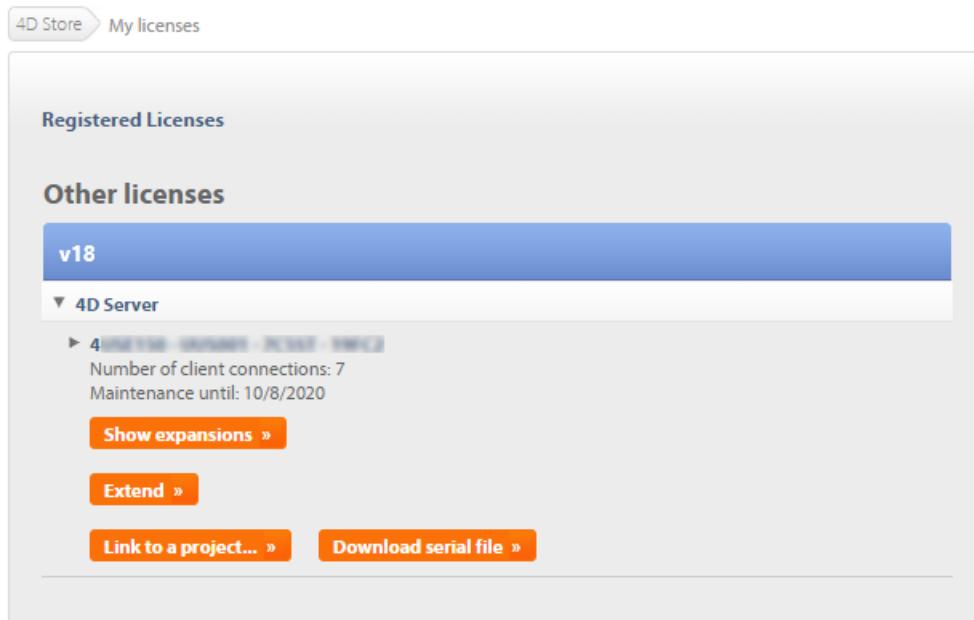
Después de iniciar sesión, puede hacer clic en Lista de licencias en la parte superior derecha de la página:

## MY LICENSES

[License list »](#)  
[License Registration »](#)  
[Purchase an Upgrade »](#)  
[Upgrade Under Maintenance »](#)

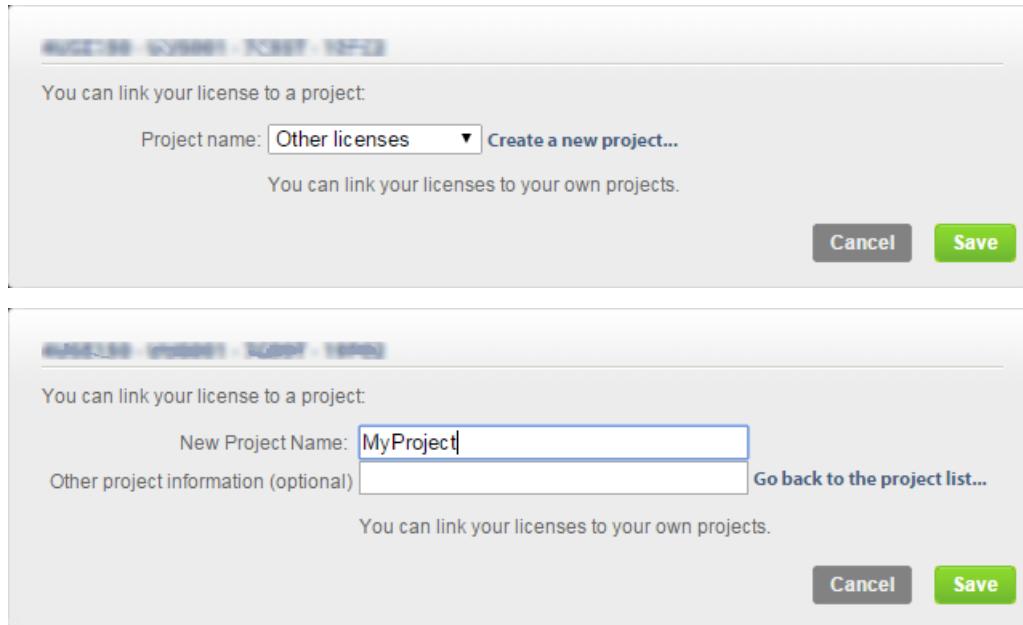
Aquí puede gestionar sus licencias asignándolas a proyectos.

Seleccione la licencia adecuada de la lista y, a continuación, haga clic en \*\*Enlazar con un proyecto...> \*\*:



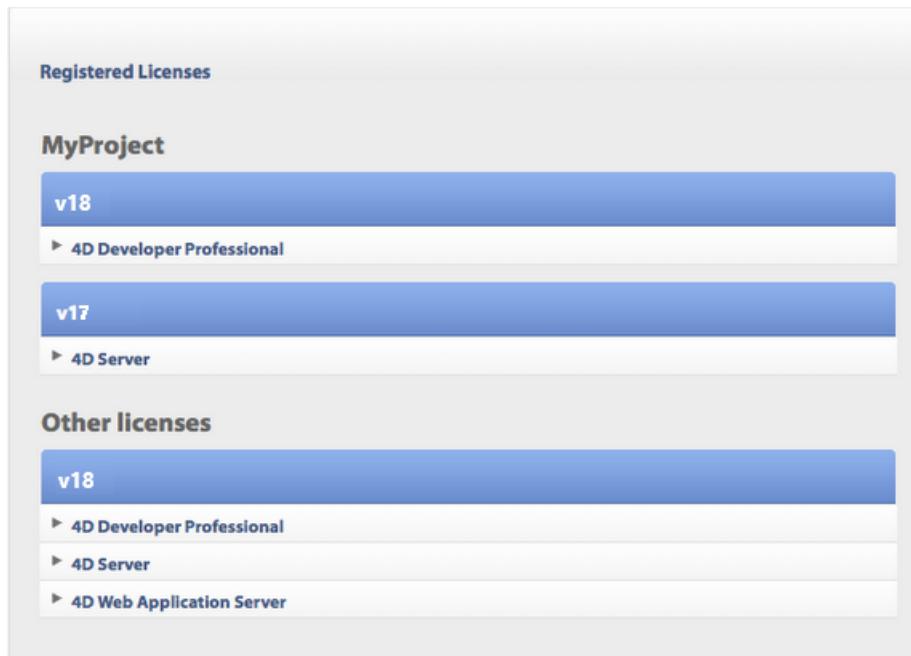
The screenshot shows the 'My licenses' interface. At the top left is a '4D Store' button and a 'My licenses' link. Below is a section titled 'Registered Licenses' containing a list for 'v18'. One item in the list is '4D Server' with the following details: 'Number of client connections: 7' and 'Maintenance until: 10/8/2020'. Below the list are three buttons: 'Show expansions »', 'Extend »', and 'Link to a project... »' (highlighted in orange), and 'Download serial file »'.

Puede seleccionar un proyecto existente o crear uno nuevo:



The first screenshot shows the 'Link to a project...' dialog with a dropdown menu set to 'Other licenses' and a 'Create a new project...' link. The second screenshot shows the same dialog with a 'New Project Name:' field containing 'MyProject' and a 'Go back to the project list...' link.

Puede utilizar los proyectos para organizar sus licencias según sus necesidades:



## Solución de problemas

Si el proceso de instalación o activación falla, compruebe la siguiente tabla, en la que se indican las causas más comunes de mal funcionamiento:

Síntomas	Causas posibles	Solución(es)
Imposible descargar el producto desde el sitio web de 4D	Sitio de Internet no disponible, aplicación antivirus, cortafuegos	1- Inténtelo de nuevo más tarde O 2- Desactive temporalmente su aplicación antivirus o su cortafuegos.
Imposible instalar el producto en el disco (instalación rechazada).	Derechos de acceso de usuario insuficientes	Abra una sesión con derechos de acceso que le permitan instalar aplicaciones (acceso administrador)
Fallo de activación en línea	Aplicación antivirus, cortafuegos, proxy	1- Desactivar temporalmente su aplicación antivirus o su cortafuegos O 2- Utilizar la activación diferida (no disponible con las licencias de las versiones "R")

Si esta información no le ayuda a resolver su problema, contacte 4D o a su distribuidor local.

## Contactos

Para cualquier pregunta sobre la instalación o activación de su producto, póngase en contacto con 4D, Inc. o con su distribuidor local.

Para US:

- Web: <https://us.4d.com/4d-technical-support>
- Tel: 1-408-557-4600

Para UK:

- Web: <https://uk.4d.com/4d-technical-support>
- Teléfono: 01625 536178

# Gestión de usuarios y grupos 4D

En las aplicaciones multiusuarios, 4D ofrece a los usuarios ciertos privilegios de acceso estándar y ciertas prerrogativas. Una vez iniciado un sistema de usuarios y grupos, estos privilegios estándar toman efecto.

## Usuarios y grupos en los proyectos

En las aplicaciones proyecto (archivos .4DProject o .4dz), los usuarios y grupos 4D pueden configurarse tanto en entornos monopuesto como en multiusuarios. Sin embargo, el control de acceso sólo es efectivo con 4D Server. La siguiente tabla enumera las principales funcionalidades de los usuarios y grupos y su disponibilidad:

	4D (monopuesto)	4D Server
Añadir/editar usuarios y grupos	sí	sí
Asignar el acceso de usuarios/grupos a los servidores	sí	sí
Identificación del usuario	no (todos los usuarios son Diseñador)	sí
Control de acceso una vez que se ha asignado una contraseña al Diseñador	no (todos los accesos son Diseñador)	sí

Para obtener información sobre la identificación de usuarios y el control de acceso en los despliegues monopuesto, consulte [este párrafo](#).

## Diseñador y administrador

El usuario más poderoso se llama el Diseñador. Ningún aspecto de la aplicación es inaccesible al Diseñador. El diseñador puede:

- acceder a todos los servidores de la aplicación sin restricciones,
- crear usuarios y grupos,
- asignar privilegios de acceso a los grupos,
- acceder al entorno Diseño. En un entorno monopuesto, siempre se utilizan los derechos de acceso del Diseñador. En el entorno cliente/servidor, la asignación de una contraseña al Diseñador activa la visualización del diálogo de inicio de sesión del usuario 4D. El acceso al entorno Diseño es de sólo lectura.

Después del Diseñador, el siguiente usuario más poderoso es el Administrador, al que se le suelen encomendar las tareas de gestión del sistema de acceso y las funciones de administración.

El administrador puede:

- crear usuarios y grupos,
- acceder al monitor y a la ventana de administración de 4D Server
- acceder a la ventana CSM para supervisar la copia de seguridad, la restauración o el servidor.

El administrador no puede:

- modificar el usuario Diseñador
- por defecto, el acceso a las partes protegidas de la aplicación. En particular, el administrador no puede acceder al modo Diseño si está restringido. El Administrador debe formar parte de uno o más grupos para tener privilegios de acceso en la aplicación. El administrador se coloca en todos los grupos nuevos, pero puede eliminar el nombre del administrador de cualquier grupo.

Tanto el Diseñador como el Administrador están disponibles por defecto en todas las aplicaciones. En la [caja de diálogo de gestión de usuarios](#), los iconos del Diseñador y del Administrador se muestran en rojo y verde respectivamente:

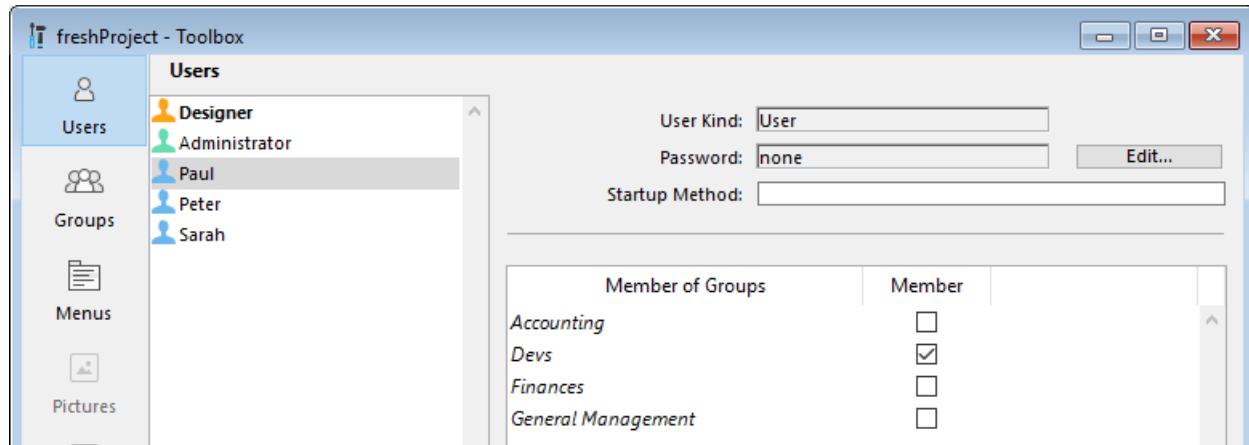
- Icône Designer:
- Icône de l'Administrateur:

Puede cambiar el nombre de los usuarios Diseñador y Administrador. En el lenguaje, el ID del diseñador es siempre 1 y el ID del administrador es siempre 2.

El Diseñador y el Administrador pueden crear hasta 16.000 grupos y 16.000 usuarios cada uno.

## Editor de usuarios

El editor de usuarios se encuentran en la caja de herramientas de 4D.



El editor de usuarios y grupos se puede mostrar en tiempo de ejecución utilizando el comando [EDIT ACCESS](#). Toda la configuración de usuarios y grupos también puede editarse durante la ejecución de la aplicación utilizando los comandos del lenguaje 4D del tema [Usuarios y Grupos](#).

## Añadir y modificar usuarios

El editor de usuarios permite crear cuentas de usuario, definir sus propiedades y asignarlas a distintos grupos.

Para añadir un usuario desde la caja de herramientas :

1. Seleccione Caja de herramientas > Usuarios en el menú Diseño o haga clic en el botón Caja de herramientas de la barra de herramientas de 4D. 4D muestra el editor de usuarios.

La lista de usuarios muestra todos los usuarios, incluyendo el [Diseñador y el Administrador](#).

2. Haga clic en el botón situado debajo de la lista de usuarios. O Haga clic derecho en la lista de usuarios y seleccione Añadir o Duplicar en el menú contextual.

El comando Duplicar se puede utilizar para crear varios usuarios que tengan las mismas características rápidamente.

4D añade un nuevo usuario a la lista, llamado por defecto "Nuevo usuarioX".

3. Introduzca el nombre de usuario. Este nombre será utilizado por el usuario para abrir la aplicación. Puede renombrar un usuario en cualquier momento utilizando el comando Renombrar del menú contextual, o utilizando los atajos Alt+clic (Windows) u Opción+clic (macOS), o haciendo dos veces clic en el nombre que quiera cambiar.
4. Para introducir una contraseña del usuario, haga clic en el botón Editar... en el área de propiedades del usuario e introduzca la contraseña dos veces en la caja de diálogo. Puede utilizar hasta 15 caracteres alfanuméricos para una contraseña. El editor de contraseñas es sensible a las mayúsculas y minúsculas.

Los usuarios pueden cambiar su contraseña en cualquier momento según las opciones de la página "Seguridad" de las propiedades de la estructura, o utilizando el comando [CHANGE PASSWORD](#).

5. Defina el grupo o los grupos a los que pertenece el usuario mediante la tabla "Miembro de los grupos". Puede añadir

o eliminar el usuario seleccionado a un grupo marcando la opción correspondiente en la columna Miembro.

La pertenencia de los usuarios a los distintos grupos también puede definirse por grupos en la página [Grupos](#).

## Eliminar un usuario

Para eliminar un usuario, selecciónelo y haga clic en el botón de eliminación o utilice el comando Suprimir del menú contextual.



Los nombres de usuario borrados ya no aparecen en el editor de usuarios. Tenga en cuenta que los ID de los usuarios eliminados se reasignan cuando se crean nuevas cuentas de usuario.

## Propiedades de los usuarios

- El campo Tipo de usuario: el campo Tipo de usuario contiene "Diseñador", "Administrador" o (para todos los demás usuarios) "Usuario".
- Método de inicio: nombre de un método asociado que se ejecutará automáticamente cuando el usuario abra la aplicación (opcional). Este método puede utilizarse, por ejemplo, para cargar las preferencias usuario.

## Editor de grupos

El editor de grupos se encuentra en la caja de herramientas de 4D.

### Configurar grupos

El editor de grupos sirve para definir los elementos que contiene cada grupo (usuarios y/o otros grupos) y para distribuir los accesos a los plug-ins.

Tenga en cuenta que una vez que se ha creado un grupo, no se puede eliminar. Si desea desactivar un grupo, sólo tiene que eliminar los usuarios que contiene.

Para crear un grupo:

1. Seleccione Caja de herramientas > Grupos en el menú Diseño o haga clic en el botón Caja de herramientas de la barra de herramientas de 4D luego haga clic en el botón Grupos. 4D muestra la ventana del editor de grupos. La lista de grupos muestra todos los grupos del proyecto de aplicación.
2. Haga clic en el botón ubicado debajo de la lista de grupos.  
O  
Haga clic derecho en la lista de grupos y elija el comando Añadir o Duplicar en el menú contextual.

El comando Duplicar se puede utilizar para crear varios grupos que tengan las mismas características rápidamente.

4D añade un nuevo grupo a la lista, llamado por defecto "Nuevo grupoX".

3. Introduzca el nombre del nuevo grupo. El nombre del grupo puede tener hasta 15 caracteres. Puede renombrar un grupo en cualquier momento utilizando el comando Renombrar del menú contextual, o utilizando los atajos Alt+clic (Windows) u Opción+clic (macOS), o haciendo dos veces clic en el nombre que quiera cambiar.

### Colocar los usuarios o los grupos en grupos

Puede colocar cualquier usuario o grupo en un grupo, y también puede colocar el propio grupo en varios otros grupos. No es obligatorio colocar a un usuario en un grupo.

Para incluir a un usuario o grupo en un grupo, basta con marcar la opción "Miembro" para cada usuario o grupo en el área de atribución de los miembros:

	User / Group	Member
	Administrator	<input checked="" type="checkbox"/>
	Designer	<input type="checkbox"/>
	New user	<input type="checkbox"/>
	Paul	<input type="checkbox"/>
	Peter	<input checked="" type="checkbox"/>
	Sarah	<input type="checkbox"/>
	Finances	<input checked="" type="checkbox"/>
	General Management	<input checked="" type="checkbox"/>
	Devs	<input type="checkbox"/>
	Admins	<input type="checkbox"/>

Si se marca el nombre de un usuario, éste se añade al grupo. Si marca el nombre de un grupo, todos los usuarios del grupo se añaden al nuevo grupo. El usuario o grupo afiliado tendrá entonces los mismos privilegios de acceso que los asignados al nuevo grupo.

La colocación de grupos dentro de otros grupos permite crear una jerarquía de usuarios. Los usuarios de un grupo colocados en otro grupo tendrán los privilegios de acceso de ambos grupos. Ver "[Un esquema de jerarquía de acceso](#)" abajo.

Para eliminar un usuario o grupo de otro grupo, basta con deseleccionar la opción correspondiente en el área de asignación de miembros.

## Asignar un grupo a un plug-in o a un servidor

Puede asignar privilegios de grupo a cualquier plug-in instalado en el proyecto. Esto incluye todos los plug-ins de 4D y los de terceros.

Distribuir los accesos a los plug-ins le permite controlar el uso de las licencias que posee para estos plug-ins. Todo usuario que no pertenezca al grupo de acceso de un plug-in no puede cargar este plug-in.

Las licencias utilizadas permanecen vinculadas a las cuentas de usuario 4D del grupo durante toda la sesión 4D.

El área "Plug-in" de la página Grupos de la caja de herramientas lista todos los plug-ins cargados por la aplicación 4D. Para dar acceso a un grupo a un complemento, basta con marcar la opción correspondiente.

Plug-in	Access
4D Client Web Server	<input type="checkbox"/>
4D Client SOAP Server	<input type="checkbox"/>
4D Write PRO	<input checked="" type="checkbox"/>
4D View PRO	<input type="checkbox"/>

Las líneas 4D Client Web Server y 4D Client SOAP Server permiten controlar la posibilidad de publicación Web y SOAP (Web Services) de cada 4D en modo remoto. Estas licencias son consideradas por 4D Server como licencias plug-in. Por lo tanto, al igual que en el caso de los plug-ins, puede restringir el derecho de uso de estas licencias a un grupo específico de usuarios.

## Un esquema de acceso jerárquico

La mejor manera de garantizar la seguridad de su aplicación y ofrecer a los usuarios diferentes niveles de acceso es utilizar un esquema de jerarquía de acceso. Los usuarios pueden ser asignados a los grupos apropiados y los grupos pueden ser anidados para crear una jerarquía de derechos de acceso. Esta sección describe varios enfoques de este esquema.

En este ejemplo, un usuario es asignado a uno de los tres grupos en función de su nivel de responsabilidad. Los usuarios asignados al grupo Contabilidad son responsables de la entrada de datos. Los usuarios asignados al grupo Finanzas son responsables de mantener los datos, incluyendo la actualización de los registros y la eliminación de los registros obsoletos. Los usuarios asignados al grupo de Gestión General se encargan de analizar los datos, incluyendo la realización de búsquedas y la impresión de informes analíticos.

A continuación, los grupos se anidan para que los privilegios se distribuyan correctamente entre los usuarios de cada grupo.

- The General Management group contains only “high-level” users.

User / Group	Member
Administrator	<input checked="" type="checkbox"/>
Designer	<input type="checkbox"/>
Paul	<input type="checkbox"/>
Peter	<input type="checkbox"/>
Sarah	<input type="checkbox"/>
Accounting	<input type="checkbox"/>
Finances	<input type="checkbox"/>
Devs	<input type="checkbox"/>
Admins	<input type="checkbox"/>

Plug-in	Access
4D Client Web Server	<input type="checkbox"/>
4D Client SOAP Server	<input type="checkbox"/>
4D Write PRO	<input type="checkbox"/>
4D View PRO	<input type="checkbox"/>

- The Finances group contains data maintenance users as well as General Management users, thus the users in General Management have the privileges of the Finances group as well.

User / Group	Member
Administrator	<input checked="" type="checkbox"/>
Designer	<input type="checkbox"/>
Paul	<input type="checkbox"/>
Peter	<input type="checkbox"/>
Sarah	<input type="checkbox"/>
Accounting	<input type="checkbox"/>
General Management	<input checked="" type="checkbox"/>
Devs	<input type="checkbox"/>
Admins	<input type="checkbox"/>

Plug-in	Access
4D Client Web Server	<input type="checkbox"/>
4D Client SOAP Server	<input type="checkbox"/>
4D Write PRO	<input type="checkbox"/>
4D View PRO	<input type="checkbox"/>

- The Accounting group contains data entry users as well as Finances group users, so the users who belong to the Finances group and the General Management group enjoy the privileges of the Accounting group as well.

User / Group	Member
Administrator	<input checked="" type="checkbox"/>
Designer	<input type="checkbox"/>
Paul	<input checked="" type="checkbox"/>
Peter	<input type="checkbox"/>
Sarah	<input type="checkbox"/>
Finances	<input checked="" type="checkbox"/>
General Management	<input checked="" type="checkbox"/>
Devs	<input type="checkbox"/>
Admins	<input type="checkbox"/>

Plug-in	Access
4D Client Web Server	<input type="checkbox"/>
4D Client SOAP Server	<input type="checkbox"/>
4D Write PRO	<input checked="" type="checkbox"/>
4D View PRO	<input type="checkbox"/>

Puede decidir qué privilegios de acceso asignar a cada grupo en función del nivel de responsabilidad de los usuarios que contiene.

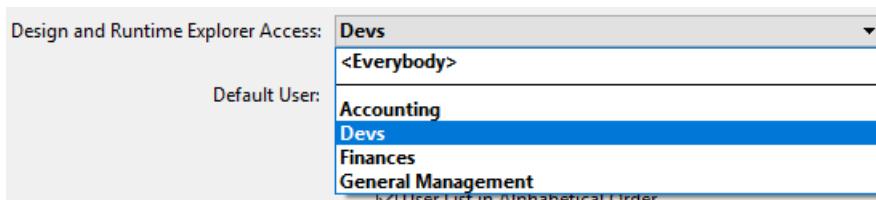
Este sistema jerárquico permite recordar fácilmente a qué grupo debe asignarse un nuevo usuario. Sólo tiene que asignar cada usuario a un grupo y utilizar la jerarquía de grupos para determinar los accesos.

## Asignación de acceso a grupos

Los grupos tienen asignados privilegios de acceso para partes o funcionalidades específicas de la aplicación:

- Acceso al entorno Diseño y al Explorador de ejecución,
- Servidor HTTP,
- Servidor REST,
- Servidor SQL.

Estos accesos se definen en la caja de diálogo Parámetros. El siguiente ejemplo muestra los derechos de acceso del explorador de diseño y tiempo de ejecución asignados al grupo "Devs":



También se utilizan grupos para [distribuir las licencias disponibles](#). Esta distribución se define en el editor Grupos.

## Archivo directory.json

Los usuarios, grupos, así como sus derechos de acceso se almacenan en un archivo específico del proyecto llamado directory.json.

Este archivo puede ser almacenado en las siguientes ubicaciones, dependiendo de sus necesidades:

- Si desea utilizar el mismo directorio para todos los archivos de datos (o si utiliza un único archivo de datos), almacene el archivo directory.json en la carpeta de configuración del usuario, es decir, en la carpeta "Settings" en el [mismo nivel que la carpeta "Project"](#) (ubicación predeterminada).
- Si desea utilizar un archivo directorio específico para archivar datos, almacene el archivo directory.json en la carpeta "[Settings](#)" de la carpeta "[Data](#)". Si un archivo directory.json está presente en esta ubicación, tiene prioridad sobre el archivo en la carpeta Settings usuario. Esta configuración personalizada/local de los usuarios y de los grupos no se verá afectada por una actualización de la aplicación.

Para permitir cambios seguros de contraseñas y pertenencias a grupos en un entorno desplegado, puede incluir su archivo directory.json en la aplicación del servidor durante la creación, utilizando la opción [correspondiente a la aplicación de creación](#).

# Página de información

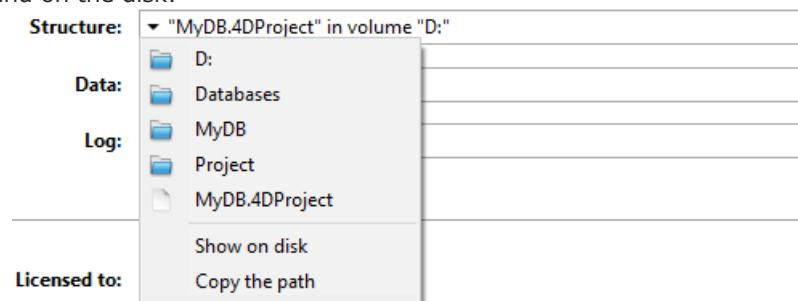
La página Información proporciona información sobre los entornos 4D y sistema, así como sobre los archivos de la base de datos y de la aplicación. Cada página puede visualizarse mediante los controles de pestañas en la parte superior de la ventana.

## Programa

Esta página indica el nombre, la versión y la ubicación de la aplicación, así como la carpeta 4D activa (para más información sobre la carpeta 4D activa, consulte la descripción del comando `Get 4D folder` en el manual *Lenguaje 4D*).

The central part of the window indicates the name and location of the project and data files as well as the log file (if any). The lower part of the window indicates the name of the 4D license holder, the type of license, and the name of the current 4D user.

- Display and selection of pathnames: On the Program tab, pathnames are displayed in pop-up menus containing the folder sequence as found on the disk:



If you select a menu item (disk or folder), it is displayed in a new system window. El comando Copiar la ruta copia el nombre completo de la ruta en el portapapeles como texto, utilizando los separadores de la plataforma actual.

- Carpeta "Licenses": El botón Carpeta "Licenses" muestra el contenido de la carpeta Licenses activa en una nueva ventana sistema. Todos los archivos de licencia instalados en su entorno 4D están agrupados en esta carpeta, en su disco duro. Cuando se abren con un navegador web, estos archivos muestran información sobre las licencias que contienen y sus características. La ubicación de la carpeta "Licenses" puede variar en función de la versión de su sistema operativo. Para obtener más información sobre la ubicación de esta carpeta, consulte el comando `Get 4D folder`. Nota: también puede acceder a esta carpeta desde la caja de diálogo "Actualizar licencias" (disponible en el menú Ayuda).

## Tablas

Esta página ofrece una visión general de las tablas de su base:

Maintenance and Security Center

Information

Program Tables Data

ID	Tables	Records	Fields	Indexes	Encryptable	Encrypted	Address Table Size
1	Employee	5 083	3	1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	5 083
2	Company	2 520	2	1	<input type="checkbox"/>	<input type="checkbox"/>	2 520
3	Cities	2 575	2	0	<input type="checkbox"/>	<input type="checkbox"/>	2 575
Total		3	10 178	7	2	1	10 178

La información de esta página está disponible tanto en el modo estándar como en el de mantenimiento.

La página lista todas las tablas de la base (incluidas las tablas invisibles), así como sus características:

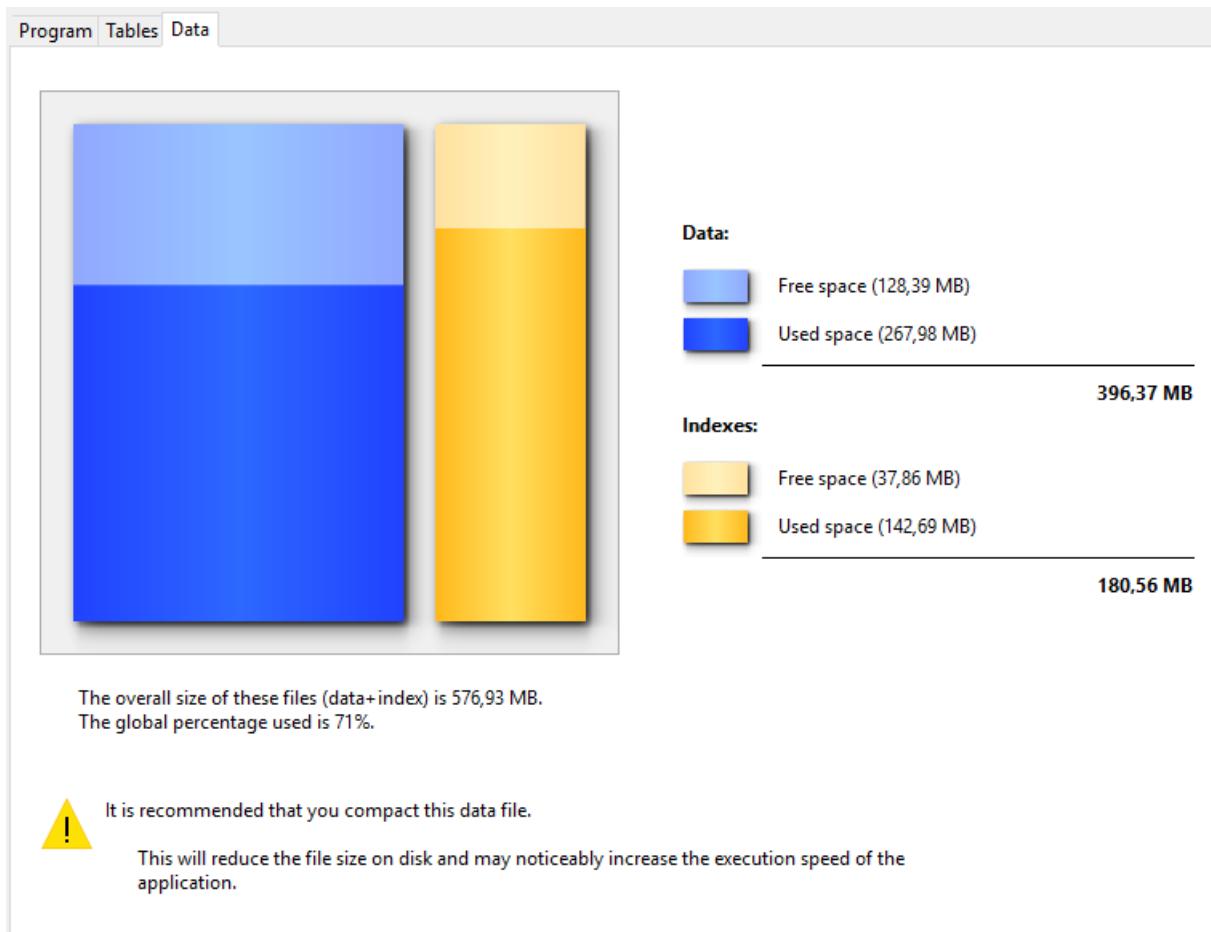
- ID: número internos de la tabla.
  - Tablas: nombres de las tablas. Los nombres de las tablas borradas se muestran entre paréntesis (si todavía están en la papelera).
  - Registros: número total de registros en la tabla. Si un registro está dañado o no se puede leer, se muestra *Error* en lugar del número. En este caso, puede considerar el uso de las herramientas de verificación y de reparación.
  - Campos: número de campos en la tabla. Los campos invisibles se cuentan, sin embargo, los campos borrados no se cuentan.
  - Índices: número de índices de todo tipo en la tabla
  - Encryptable: If checked, the Encryptable attribute is selected for the table at the structure level (see "Encryptable" paragraph in the Design Reference Manual).
  - Encriptado: si se marca, los registros de la tabla se cifran en el archivo de datos. *Note: Any inconsistency between Encryptable and Encrypted options requires that you check the encryption status of the data file in the Encrypt page of the MSC.*
  - Tamaño tabla direcciones: tamaño de la tabla de direcciones para cada tabla. La tabla de direcciones es una tabla interna que almacena un elemento por cada registro creado en la tabla. De hecho, vincula los registros a su dirección física. Por razones de rendimiento, no se redimensiona cuando se eliminan registros, por lo que su tamaño puede ser diferente del número de registros actual de la tabla. Si esta diferencia es significativa, se puede ejecutar una operación de compactación de datos con la opción "Compactar la tabla de direcciones" marcada para optimizar el tamaño de la tabla de direcciones (ver la página [Compactar](#)). *Nota: las diferencias entre el tamaño de la tabla de direcciones y el número de registros también pueden ser el resultado de un incidente durante la escritura de la caché en el disco.*

## Datos

La página Datos ofrece información sobre el espacio de almacenamiento disponible y utilizado en el archivo de datos.

No se puede acceder a esta página en modo mantenimiento

La información se ofrece en forma gráfica:



Esta página no tiene en cuenta los datos que puedan estar almacenados fuera del archivo de datos (ver "Almacenamiento externo").

Los archivos demasiado fragmentados reducen el rendimiento del disco y, por tanto, de la base. Si la tasa de ocupación es demasiado baja, 4D lo indicará con un icono de advertencia (que aparece en el botón de información y en la pestaña del tipo de archivo correspondiente) y especificará que es necesario compactar:



También aparece un icono de advertencia en el botón de la página Compactar:



# Página Análisis de actividades

La página Análisis de actividades permite ver el contenido del archivo de registro actual. This function is useful for parsing the use of an application or detecting the operation(s) that caused errors or malfunctions. In the case of an application in client-server mode, it allows verifying operations performed by each client machine.

También es posible revertir las operaciones realizadas sobre los datos de la base. Para más información, consulte [Página de retroceso](#).

The screenshot shows the 'Maintenance and Security Center' interface. On the left, there is a sidebar with icons for various maintenance tasks: Information, Activity analysis (which is selected), Verify, Backup, Compact, Rollback, Restore, Repair, and Encrypt. The main area is titled 'Activity analysis' and contains a table with the following columns: Operation /91, Action Sequence, Table Parts, Primary..., Process 20, Size, Date 10/12/2019 16:37, Hour 10:37, System User aschmitt, 4D User main user, Values 199 ; ; ; , and Record 197. The table lists numerous operations performed on parts and invoices, such as additions and sequences, with dates ranging from 10/12/2019 to 11/12/2019 and times from 16:37 to 16:37. At the bottom of the table are navigation arrows and buttons for Analyze, Browse, and Export... .

Cada operación registrada en el archivo de registro aparece como una línea. Las columnas ofrecen información variada sobre la operación. Puede reorganizar las columnas como desee haciendo clic en sus encabezados.

Esta información permite identificar la fuente y el contexto de cada operación:

- Operación: número de secuencia de la operación en el archivo de historial.
- Acción: tipo de operación realizada sobre los datos. Esta columna puede contener una de las siguientes operaciones:
  - Apertura del archivo de datos: apertura de un archivo de datos.
  - Cierre del archivo de datos: cierre de un archivo de datos abierto.
  - Creación de un contexto: creación de un proceso que especifica un contexto de ejecución.
  - Cierre de un contexto: cierre de un proceso.
  - Adición: creación y almacenamiento de un registro.
  - Añadir un BLOB: almacenamiento de un BLOB en un campo BLOB.
  - Eliminación: eliminación de un registro.
  - Modificación: modificación de un registro.

- Inicio de la transacción: transacción iniciada.
- Validación de transacción: transacción validada.
- Cancelación de transacción: transacción cancelada.
- Update context: Change in extra data (e.g. a call to `CHANGE CURRENT USER` or `SET USER ALIAS` ).
- Tabla: tabla a la que pertenece el registro añadido/borrado/modificado o el BLOB.
- Llave primaria/BLOB: contenido de la llave primaria de cada registro (cuando la llave primaria se compone de varios campos, los valores se separan con punto y coma) o número de secuencia del BLOB implicado en la operación.
- Proceso: número interno del proceso en el que se realizó la operación. Este número interno corresponde al contexto de la operación.
- Tamaño: tamaño (en bytes) de los datos procesados por la operación.
- Fecha y hora: fecha y hora en que se realizó la operación.
- System User: System name of the user that performed the operation. In client-server mode, the name of the client-side machine is displayed; in single-user mode, the session name of the user is displayed.
- 4D User: 4D user name of the user that performed the operation. If an alias is defined for the user, the alias is displayed instead of the 4D user name.
- Valores: valores de los campos del registro en caso de adición o de modificación. Los valores están separados por ";". Sólo se muestran los valores representados en forma alfanumérica.  
*Nota: si la base está encriptada y no se ha ofrecido una llave de datos válida correspondiente al archivo de historial abierto, los valores encriptados no se muestran en esta columna.*
- Registros: número del registro.

Click on Analyze to update the contents of the current log file of the selected application (named by default `dataname.journal`). The Browse button can be used to select and open another log file for the application. El botón Exportar... puede utilizarse para exportar el contenido del archivo como texto.

# Página Verificación

Esta página se utiliza para verificar la integridad de los datos. La verificación puede llevarse a cabo en los registros y/o índices. Esta página sólo comprueba la integridad de los datos. Si se encuentran errores y es necesario repararlos, se le indicará que utilice la página [Reparación](#).

## Acciones

La página contiene botones de acción que dan acceso directo a las funciones de verificación.

Cuando la base está encriptada, la verificación incluye la validación de la consistencia de los datos encriptados. Si no se ha suministrado una llave de datos válida, aparecerá un diálogo solicitando la frase secreta o la llave de datos.

- Verificar los registros y los índices: inicia el procedimiento de verificación total de los datos.
- Verificar sólo los registros: inicia el procedimiento de verificación sólo para los registros (los índices no se verifican).
- Verificar sólo los índices: inicia el procedimiento de verificación sólo para los índices (los registros no se verifican).

Verification of records and indexes can also be carried out in detail mode, table by table (see the Details section below).

## Abrir archivo de historial

Regardless of the verification requested, 4D generates a log file in the `Logs` folder of the application. Este archivo lista todas las verificaciones realizadas e indica los errores encontrados, cuando sea el caso (se muestra [OK] cuando la verificación es correcta). It is created in XML format and is named: `ApplicationNameVerify_Logyyyy-mm-dd hh-mm-ss.xml` where:

- `ApplicationName` is the name of the project file without any extension, for example "Invoices",
- `yyyy-mm-dd hh-mm-ss` es la marca de tiempo del archivo, basada en la hora del sistema local cuando se inició la operación de mantenimiento, por ejemplo "2019-02-11 15-20-45".

Al presionar el botón Abrir archivo de historial, 4D muestra el archivo de historial más reciente en el navegador por defecto de la máquina.

## Detalles

El botón Lista de tablas muestra una página detallada que puede utilizarse para ver y seleccionar los registros e índices reales que se van a verificar:

	Tables	Action	Status
Employee	Verify indexes	[Status icon]	
Records	(7324866 record(s))	[Status icon]	
Indexed fields	(1 indexed field(s))	[Status icon]	
Employee.ID		[Status icon]	
Company	Verify indexes	[Status icon]	
Records	(7322302 record(s))	[Status icon]	
Indexed fields	(1 indexed field(s))	[Status icon]	
Company.ID		[Status icon]	
Cities		[Status icon]	
Records	(2073735 record(s))	[Status icon]	

Some records (2 073 735 out of 16 720 903) will be verified

All indexes (2) will be verified

**Verify**

La designación de los elementos a verificar le permite ahorrar tiempo durante el procedimiento de verificación.

La lista principal muestra todas las tablas de la base. Para cada tabla, puede limitar la verificación a los registros y/o índices. Despliegue el contenido de una tabla o los campos indexados y marque/desmarque las casillas de selección como desee. Por defecto, todo está seleccionado. También puede utilizar los botones de acceso directo Seleccionar todo, Deseleccionar todo, Todos los registros y Todos los índices.

Para cada línea de tabla, la columna "Acción" indica las operaciones a realizar. Cuando se expande la tabla, las filas "Registros" y "Campos indexados" indican el número de elementos en cuestión.

La columna "Estado" muestra el estado de verificación de cada elemento mediante símbolos:

	Verificación realizada sin problemas
	Verificación efectuada, problemas encontrados
	Verificación realizada parcialmente
	Verificación no efectuada

Haga clic en Verificar para comenzar la verificación o en Estándar para volver a la página estándar.

El botón Abrir archivo de historial puede utilizarse para mostrar el archivo de historial en el navegador predeterminado de la máquina (ver [Abrir archivo de historial](#) arriba).

La página estándar no tendrá en cuenta las modificaciones realizadas en la página detallada: cuando se presiona un botón de verificación en la página estándar, se verifican todos los elementos. Por otra parte, los parámetros establecidos en la página detallada se conservan de una sesión a otra.



# Página compactado

Esta página permite acceder a las funciones de compactación del archivos de datos.

## ¿Por qué compactar los archivos?

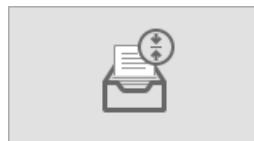
La compactación de archivos responde a dos tipos de necesidades:

- Reducción del tamaño y optimización de los archivos : los archivos pueden contener espacios no utilizados ("huecos"). De hecho, cuando se eliminan los registros, el espacio que ocupaban anteriormente en el archivo queda vacío. 4D reutiliza estos espacios vacíos siempre que es posible, pero como el tamaño de los datos es variable, las sucesivas eliminaciones o modificaciones generarán inevitablemente espacio inutilizable para el programa. Lo mismo ocurre cuando se acaba de borrar una gran cantidad de datos: los espacios vacíos quedan sin asignar en el archivo. La relación entre el tamaño del archivo de datos y el espacio realmente utilizado para los datos es la tasa de ocupación de los datos. Una tasa demasiado baja puede provocar, además de un desperdicio de espacio, el deterioro del rendimiento de la base. La compactación puede utilizarse para reorganizar y optimizar el almacenamiento de los datos con el fin de eliminar los "huecos". El área "Información" resume los datos relativos a la fragmentación de los archivos y sugiere las operaciones a realizar. La pestaña [Datos](#) de la página "Información" del CSM indica la fragmentación del archivo de datos actual.
- Actualización completa de los datos aplicando el formato actual definido en el archivo de estructura. Esto es útil cuando los datos de una misma tabla se almacenan en diferentes formatos, por ejemplo, después de un cambio en la estructura de la base.

La compactación sólo está disponible en el modo mantenimiento. If you attempt to carry out this operation in standard mode, a warning dialog box will inform you that the application will be closed and restarted in maintenance mode. You can compact a data file that is not opened by the application (see [Compact records and indexes](#) below).

## Compactación estándar

Para iniciar directamente la compactación del archivo de datos, haga clic en el botón de compactación de la ventana del CSM.



Como la compactación implica la duplicación del archivo original, el botón se desactiva cuando no hay espacio suficiente en el disco que contiene el archivo.

Esta operación compacta el archivo principal y los archivos de índice. 4D copia los archivos originales y los coloca en una carpeta llamada Archivos Reemplazados (Compactando), que se crea junto al archivo original. Si ha realizado varias operaciones de compactación, se crea una nueva carpeta cada vez. Se llamará "Archivos reemplazados (compactando)\_1", "Archivos reemplazados (compactando)\_2", y así sucesivamente. Puede modificar la carpeta donde se guardan los archivos originales utilizando el modo avanzado.

Una vez finalizada la operación, los archivos compactados sustituyen automáticamente a los originales. The application is immediately operational without any further manipulation.

Cuando la base está encriptada, la compactación incluye pasos de encriptación y desencriptación y, por tanto, requiere la llave de encriptación de datos actual. Si no se ha suministrado una llave de datos válida, aparecerá una caja de diálogo solicitando la frase secreta o la llave de datos.

Atención: cada operación de compactación implica la duplicación del archivo original, lo que aumenta el tamaño de la carpeta de la aplicación. Es importante tener esto en cuenta (especialmente en macOS, donde las aplicaciones 4D aparecen como paquetes) para que el tamaño de la aplicación no aumente excesivamente. Eliminar manualmente las copias del archivo original dentro del paquete puede ser útil para mantener el tamaño del paquete.

## Abrir archivo de historial

After compacting is completed, 4D generates a log file in the Logs folder of the project. Este archivo permite ver todas las operaciones realizadas. It is created in XML format and named: *ApplicationName\_Compact\_Log\_yyyy-mm-dd hh-mm-ss.xml*" where:

- *ApplicationName* is the name of the project file without any extension, for example "Invoices",
- *yyyy-mm-dd hh-mm-ss* es la marca de tiempo del archivo, basada en la hora del sistema local cuando se inició la operación de mantenimiento, por ejemplo "2019-02-11 15-20-45".

Al presionar el botón Abrir archivo de historial, 4D muestra el archivo de historial más reciente en el navegador por defecto de la máquina.

## Modo avanzado

La página Compactar contiene un botón Avanzado>, con el que se puede acceder a una página de opciones para compactar el archivo de datos.

### Compactar los registros y los índices

El área Compactar los registros y los índices muestra el nombre de la ruta del archivo de datos actual, así como un botón [...] que puede utilizarse para especificar otro archivo de datos. Al hacer clic en este botón, se muestra una caja de diálogo estándar de apertura de documentos para que pueda designar el archivo de datos a compactar. Debe seleccionar un archivo de datos que sea compatible con el archivo de estructura abierto. Una vez validada esta caja de diálogo, se indica en la ventana el nombre de la ruta del archivo a compactar.

El segundo botón [...] se puede utilizar para especificar otra ubicación para guardar los archivos originales antes de la operación de compactación. Esta opción se puede utilizar más particularmente cuando se compactan archivos voluminosos mientras se utilizan diferentes discos.

### Forzar la actualización de los registros

Cuando esta opción está marcada, 4D reescribe cada registro de cada tabla durante la operación de compactación, según su descripción en la estructura. Si esta opción no está marcada, 4D sólo reorganiza el almacenamiento de datos en el disco. Esta opción es útil en los siguientes casos:

- Cuando los tipos de campo se modifican en la estructura de la aplicación después de haber introducido los datos. Por ejemplo, puede haber cambiado un campo Longint a un tipo Real. 4D permite incluso cambios entre dos tipos muy diferentes (con riesgo de pérdida de datos), por ejemplo, un campo Real puede cambiarse a Texto y viceversa. En este caso, 4D no convierte los datos ya introducidos de forma retroactiva; los datos se convierten sólo cuando se cargan los registros y luego se guardan. Esta opción obliga a convertir todos los datos.
- Cuando una opción de almacenamiento externo para datos de Texto, Imagen o BLOB ha sido cambiada después de haber introducido los datos. Esto puede ocurrir cuando se convierten las bases de datos desde una versión anterior a la v13. Como en el caso de la reescritura descrita anteriormente, 4D no convierte los datos ya introducidos con carácter retroactivo. Para ello, puede forzar la actualización de los registros para aplicar el nuevo modo de almacenamiento a los registros ya introducidos.
- Cuando se han eliminado las tablas o los campos. En este caso, la compactación con actualización de registros recupera el espacio de estos datos eliminados y reduce así el tamaño del archivo.

Todos los índices se actualizan cuando se selecciona esta opción.

### Compactar la tabla de direcciones

(opción activa únicamente cuando la opción anterior está marcada)

Esta opción reconstruye completamente la tabla de direcciones para los registros durante la compactación. Esto optimiza el tamaño de la tabla de direcciones y se utiliza principalmente para las bases de datos en las que se crearon grandes volúmenes de datos y luego se borraron. En otros casos, la optimización no es un factor decisivo.

Tenga en cuenta que esta opción ralentiza sustancialmente la compactación e invalida cualquier conjunto guardado mediante el comando `SAVE SET`. Además, recomendamos especialmente que se borren los conjuntos guardados en este caso, ya que su uso puede dar lugar a selecciones de datos incorrectas.

- La compactación tiene en cuenta los registros de las tablas que se han puesto en la Papelera. Si hay un gran número de registros en la Papelera, esto puede ser un factor adicional que puede ralentizar la operación.
- El uso de esta opción hace que la tabla de direcciones, y por tanto la base de datos, sea incompatible con el archivo de diario actual (si existe). It will be saved automatically and a new journal file will have to be created the next time the application is launched.
- Puede decidir si la tabla de direcciones necesita ser compactada comparando el número total de registros y el tamaño de la tabla de direcciones en la página [Información del CSM](#).

# Página Retroceso

La página Retroceso permite acceder a la función de retorno entre las operaciones realizadas en el archivo de datos. Se asemeja a una función de anulación aplicada en varios niveles. Es especialmente útil cuando un registro ha sido borrado por error en una base de datos.

This function is only available when the application functions with a data log file.

Maintenance and Security Center

**Rollback**

The list below shows all the performed operations recorded in the log file since the last backup.

Operation	Action	Table	Primary...	Process	Size	Date	Hour	System User	4D User	Values	Record
1006	Deletion	Parts	103		63	16/12/2019	18:10	aschmitt	Main user		102
1007	Deletion	Parts	102		63	16/12/2019	18:10	aschmitt	Main user		101
1008	Deletion	Parts	101		63	16/12/2019	18:10	aschmitt	Main user		100
1009	Deletion	Parts	100		63	16/12/2019	18:10	aschmitt	Main user		99
1010	Deletion	Parts	99		63	16/12/2019	18:10	aschmitt	Main user		98
1011	Deletion	Parts	98		63	16/12/2019	18:10	aschmitt	Main user		97
1012	Deletion	Parts	97		63	16/12/2019	18:10	aschmitt	Main user		96
1013	Deletion	Parts	146		63	16/12/2019	18:10	aschmitt	Main user		145
1014	Deletion	Parts	145		63	16/12/2019	18:10	aschmitt	Main user		144
1015	Deletion	Parts	144		63	16/12/2019	18:10	aschmitt	Main user		143
1016	Deletion	Parts	143		63	16/12/2019	18:10	aschmitt	Main user		142
1017	Deletion	Parts	142		63	16/12/2019	18:10	aschmitt	Main user		141
1018	Deletion	Parts	141		63	16/12/2019	18:10	aschmitt	Main user		140
1019	Deletion	Parts	8		63	16/12/2019	18:10	aschmitt	Main user		7
1020	Deletion	Parts	7		63	16/12/2019	18:10	aschmitt	Main user		6
1021	Deletion	Parts	6		63	16/12/2019	18:10	aschmitt	Main user		5

By selecting a specific log action in the list above and clicking the Rollback button, 4D Application will close the current data file, restore and open the selected backup of the database and perform all the above logged actions including the selected one.

The rollback operation cannot be undone, but the current data file will be renamed and stored on the disk.

Current log file

Rollback

Si la base de datos está encriptada y no se ha suministrado una llave de datos válida correspondiente al archivo de registro abierto, los valores encriptados no se muestran en la columna Valores y se muestra un diálogo en el que se solicita la frase secreta o la llave de datos si se hace clic en el botón Retroceso.

El contenido y el funcionamiento de la lista de operaciones es el mismo que el de la ventana [Análisis de actividades](#).

Para realizar un retroceso entre las operaciones, seleccione la linea tras la cual deben anularse todas las operaciones. La operación de la línea seleccionada será la última conservada. Si, por ejemplo, desea cancelar una eliminación, seleccione la operación situada justo antes de ella. La operación de eliminación, así como todas las operaciones posteriores, se cancelarán.

A continuación, haga clic en el botón Retroceso. 4D le pide que confirme la operación. Si hace clic en Aceptar, los datos se restauran al estado exacto en el que se encontraban en el momento de la acción seleccionada.

El menú que se encuentra en la parte inferior de la ventana permite seleccionar un archivo de historial que se utilizará cuando se aplique la función de retorno a una base restaurada desde un archivo. En este caso, debe especificar el archivo de historial de datos correspondiente al archivo.

Así es como funciona la función de retroceso: cuando el usuario hace clic en el botón Retroceso, 4D cierra la base de datos actual y restaura la última copia de seguridad de los datos de la base. La base restaurada se abre y 4D integra las operaciones del archivo de historial de datos hasta la operación seleccionada. Si la base aún no se ha guardado, 4D se inicia con un archivo de datos vacío.

# Página Reparación

Esta página se utiliza para reparar el archivo de datos cuando se ha dañado. Generalmente, sólo utilizará estas funciones bajo la supervisión de los equipos técnicos de 4D, cuando se hayan detectado anomalías al abrir la aplicación o tras una [verificación](#).

Atención: cada operación de reparación implica la duplicación del archivo original, lo que aumenta el tamaño de la carpeta de la aplicación. Es importante tener esto en cuenta (especialmente en macOS, donde las aplicaciones 4D aparecen como paquetes) para que el tamaño de la aplicación no aumente excesivamente. Eliminar manualmente las copias del archivo original dentro del paquete puede ser útil para minimizar el tamaño del paquete.

La reparación sólo está disponible en modo mantenimiento. If you attempt to carry out this operation in standard mode, a warning dialog will inform you that the application will be closed and restarted in maintenance mode. Cuando la base está encriptada, la reparación de datos incluye pasos de encriptación y desencriptación y, por tanto, requiere la llave de encriptación de datos actual. Si no se ha suministrado ya una llave de cifrado válida, aparecerá un cuadro de diálogo solicitando la frase de paso o la llave de cifrado (ver página Cifrado).

## Archivos

### Archivo de datos a reparar

Nombre de la ruta del archivo de datos actual. El botón [...] puede utilizarse para especificar otro archivo de datos. Al hacer clic en este botón, se muestra un diálogo estándar de apertura de documentos para que pueda designar el archivo de datos a reparar. If you perform a [standard repair](#), you must select a data file that is compatible with the open project file. Si realiza una reparación [reparación por encabezados de registros](#), puede seleccionar todo archivo de datos. Una vez validado este diálogo, se indica en la ventana el nombre de la ruta del archivo a reparar.

### Carpeta de copia de seguridad de los archivos originales

Por defecto, el archivo de datos original se duplicará antes de la operación de reparación. It will be placed in a subfolder named "Replaced files (repairing)" in the application folder. El segundo botón [...] se puede utilizar para especificar otra ubicación para guardar los archivos originales antes de iniciar la reparación. Esta opción se puede utilizar más particularmente cuando se reparan archivos voluminosos mientras se utilizan diferentes discos.

### Archivos reparados

4D crea un nuevo archivo de datos vacío en la ubicación del archivo original. The original file is moved into the folder named "%Replaced Files (Repairing) date time" whose location is set in the "Original files backup folder" area (application folder by default). El archivo vacío se llena con los datos recuperados.

### Reparación estándar

Se debe elegir la reparación estándar cuando sólo están dañados unos pocos registros o índices (las tablas de direcciones están intactas). Los datos se compactan y se reparan. Este tipo de reparación sólo puede realizarse cuando los datos y el archivo de estructura coinciden.

Una vez finalizado el procedimiento de reparación, aparece la página "Reparación" del CSM. Un mensaje indica si la reparación fue exitosa. Si es así, puede abrir la aplicación inmediatamente.



### Reparación por encabezados de registros

Utilice esta opción de reparación de bajo nivel sólo cuando el archivo de datos esté gravemente dañado y después de

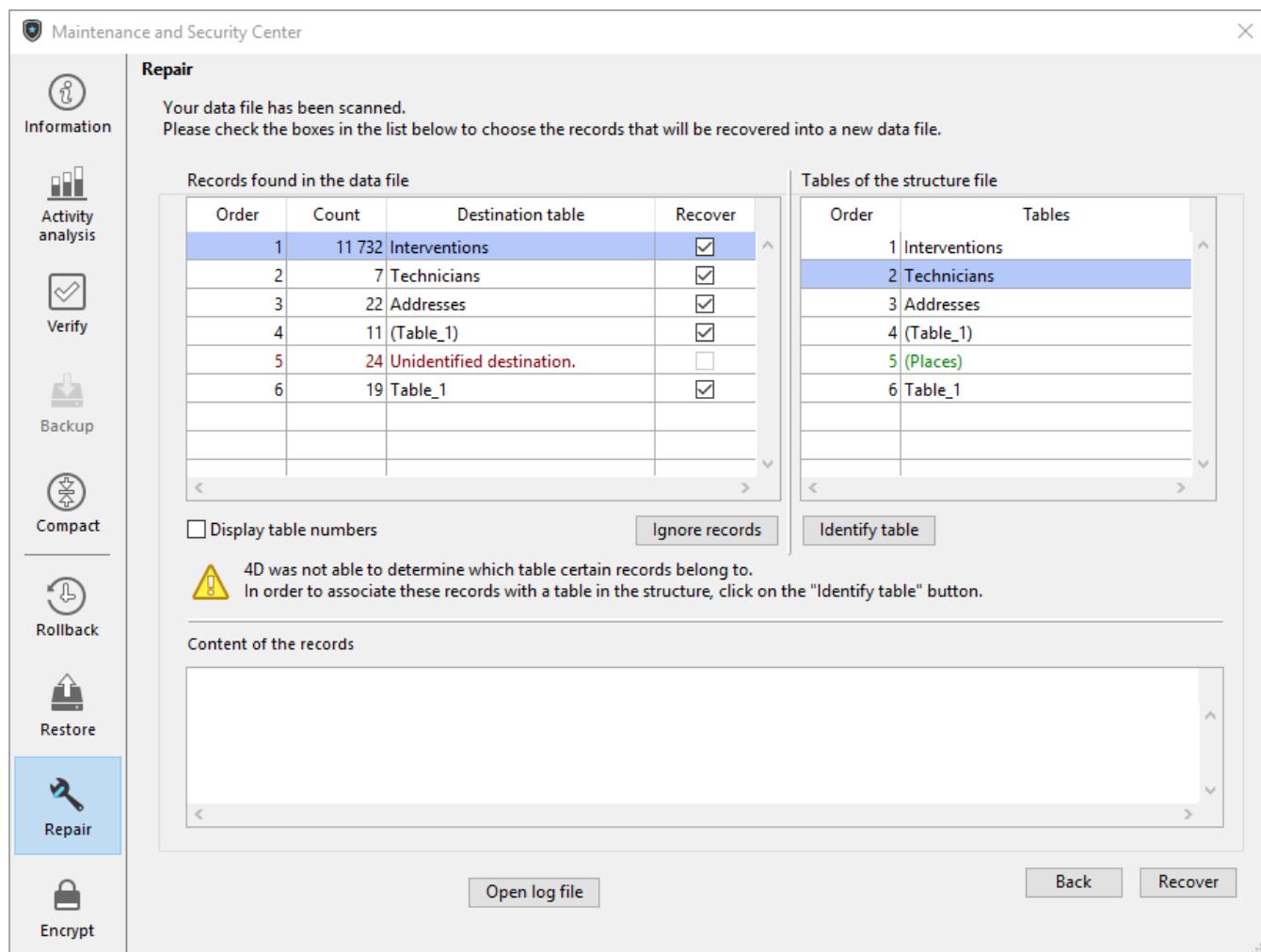
que todas las demás soluciones (restitución desde una copia de seguridad, reparación estándar) hayan resultado ineficaces.

Los registros 4D varían en tamaño, por lo que es necesario mantener la ubicación donde se almacenan en el disco en una tabla específica, llamada tabla de direcciones, para poder encontrarlos de nuevo. Por lo tanto, el programa accede a la dirección del registro por medio de un índice y de la tabla de direcciones. Si sólo están dañados los registros o los índices, la opción de reparación estándar suele ser suficiente para resolver el problema. Sin embargo, cuando la propia tabla de direcciones se ve afectada, requiere una recuperación más sofisticada, ya que será necesario reconstituirla. Para ello, el CSM utiliza el marcador situado en el encabezado de cada registro. Los marcadores se comparan con un resumen del registro, que incluye la mayor parte de su información, y a partir del cual es posible reconstruir la tabla de direcciones.

If you have deselected the Records definitively deleted option in the properties of a table in the structure, performing a recovery by header markers may cause records that were previously deleted to reappear.

La recuperación por encabezados no tiene en cuenta las restricciones de integridad. Más concretamente, tras esta operación puede obtener valores duplicados con campos únicos o valores NULL con campos declarados Nunca Null.

Al hacer clic en Escanear y reparar..., 4D realiza un análisis completo del archivo de datos. Una vez finalizado el análisis, los resultados aparecen en la siguiente ventana:



Si todos los registros y todas las tablas han sido asignados, sólo se muestra el área principal.

El área "Registros encontrados en el archivo de datos" incluye dos tablas que resumen la información del análisis del archivo de datos.

- La primera tabla lista la información del análisis del archivo de datos. Cada línea muestra un grupo de registros recuperables en el archivo de datos:

- La columna Orden indica el orden de recuperación del grupo de registros.
  - La columna Número indica el número de los registros de la tabla.
  - La columna Tabla de destino indica los nombres de las tablas que se asignaron automáticamente a los grupos de registros identificados. Los nombres de las tablas asignadas automáticamente aparecen en verde. Los grupos no asignados, es decir, las tablas a las que no se ha podido asociar ningún registro, aparecen en rojo.
  - La columna Recuperar le permite indicar, para cada grupo, si quiere recuperar los registros. Por defecto, esta opción está marcada para todos los grupos con registros que pueden asociarse a una tabla.

● La segunda tabla lista las tablas del archivo del proyecto.

## Asignación manual

Si varios grupos de registros no han podido ser asignados a las tablas debido a una tabla de direcciones dañada, puede asignarlos manualmente. Para ello, seleccione primero un grupo de registros sin asignar en la primera tabla. El área "Contenido de los registros" muestra entonces una vista previa del contenido de los primeros registros del grupo para facilitar su asignación:

The screenshot shows the 'Repair' tab in the Maintenance and Security Center. On the left, a vertical sidebar lists icons for Information, Activity analysis, Verify, Backup, Compact, Rollback, Restore, Repair (selected), and Encrypt.

**Repair Section:**

- Information:** Your data file has been scanned. Please check the boxes in the list below to choose the records that will be recovered into a new data file.
- Records found in the data file:** A table with columns Order, Count, Destination table, and Recover. It lists:
  - Order 1, Count 11,732, Destination table Interventions, Recover checked
  - Order 2, Count 7, Destination table Unidentified destination, Recover unchecked
  - Order 3, Count 22, Destination table Addresses, Recover checked
- Tables of the structure file:** A table with columns Order and Tables, listing:
  - Order 1, Tables Interventions
  - Order 2, Tables Addresses
- Buttons:** Display table numbers (checkbox), Ignore records, Identify table.
- Warning:** 4D was not able to determine which table certain records belong to. In order to associate these records with a table in the structure, click on the "Identify table" button.
- Content of the records:** A table showing fields NB01, BONAPARTE, Napoléon, Source : http://fr.wikipedia.org/wiki/Napol%C3%A9on\_Bonaparte, and others for JM01, MURAT, Joachim, Source : http://fr.wikipedia.org/wiki/Joachim\_Murat, MU01, MUIRON, Jean-Baptiste, Bien sÃºr MUIRON, colonel, and MA01, MACCENNA, Alphonse, Source : http://fr.wikipedia.org/wiki/Alphonse\_Maccenna.
- Buttons at the bottom:** Open log file, Back, Recover.

A continuación, seleccione la tabla que desea asignar al grupo en la tabla "Tablas no asignadas" y haga clic en el botón Identificar tabla. También puede asignar una tabla utilizando arrastrar y soltar. El grupo de registros se asocia entonces a la tabla y se recupera en esta tabla. Los nombres de las tablas que se asignan manualmente aparecen en negro. Utilice el botón Ignorar registros para eliminar la asociación realizada manualmente entre la tabla y el grupo de registros.

## Abrir archivo de historial

After repair is completed, 4D generates a log file in the Logs folder of the project. Este archivo permite ver todas las operaciones realizadas. It is created in XML format and named: *ApplicationName\_Repair\_Log\_**yyyy-mm-dd hh-mm-ss.xml*" where:

- *ApplicationName* is the name of the project file without any extension, for example "Invoices",

- *yyyy-mm-dd hh-mm-ss* es la marca de tiempo del archivo, basada en la hora del sistema local cuando se inició la operación de mantenimiento, por ejemplo "2019-02-11 15-20-45".

Al presionar el botón Abrir archivo de historial, 4D muestra el archivo de historial más reciente en el navegador por defecto de la máquina.

# Página de cifrado

Puede utilizar esta página para cifrar o *descifrar* (es decir, eliminar el cifrado) el archivo de datos, según el estado del atributo Encriptable definido para cada tabla de la base.

Para obtener información detallada sobre el cifrado de datos en 4D, consulte la sección "Cifrado de datos" en el manual de *Diseño*. También puede leer la entrada del blog [Una mirada más profunda al cifrado de datos en 4D](#).

Se crea una nueva carpeta cada vez que se realiza una operación de cifrado/descifrado. Se denomina "Replaced Files (Encrypting) yyyy-mm-dd hh-mm-ss> o "Replaced Files (Decrypting) yyyy-mm-dd hh-mm-ss".

El cifrado sólo está disponible en [modo mantenimiento](#). Si intenta realizar esta operación en modo estándar, un diálogo de advertencia le informará de que la aplicación se cerrará y se reiniciará en modo de mantenimiento

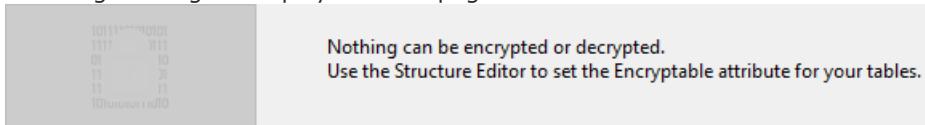
## Atención:

- Cifrar un archivo de datos es una operación de larga duración. Muestra un indicador de progreso (que puede ser interrumpido por el usuario). Note also that an application encryption operation always includes a compacting step.
- Cada operación de cifrado produce una copia del archivo de datos, lo que aumenta el tamaño de la carpeta de la aplicación. Es importante tener esto en cuenta (especialmente en macOS, donde las aplicaciones 4D aparecen como paquetes) para que el tamaño de la aplicación no aumente excesivamente. Mover o eliminar manualmente las copias del archivo original dentro del paquete puede ser útil para minimizar el tamaño del paquete.

## Cifrar datos por primera vez

Para cifrar los datos por primera vez con el CSM es necesario seguir los siguientes pasos:

1. En el editor de estructuras, marque el atributo Encriptable de cada tabla cuyos datos desee encriptar. Ver la sección "Propiedades de las tablas".
2. Abra la página de encriptación del CSM. If you open the page without setting any tables as Encryptable, the following message is displayed in the page:



Otherwise, the following message

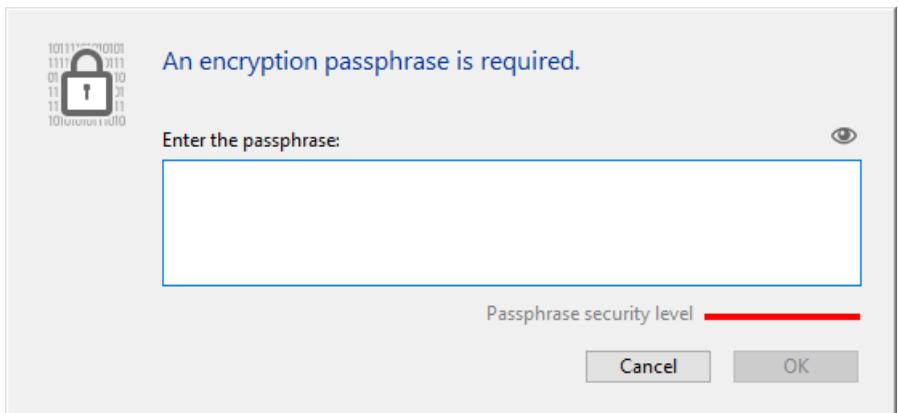


is displayed:



3. Haga clic en el botón Encriptar imagen.

Se le pedirá que introduzca una frase secreta para su archivo de datos:



La frase secreta se utiliza para

generar la llave de cifrado de los datos. Una frase secreta es una versión más segura de una contraseña y puede contener un gran número de caracteres. For example, you could enter a passphrases such as "We all came out to Montreux" or "My 1st Great Passphrase!!" For example, you could enter a passphrases such as "We all came out to Montreux" or "My 1st Great Passphrase!!" The security level indicator can help you evaluate the strength of your passphrase: (deep green is the highest level) For example, you could enter a passphrases such as "We all came out to Montreux" or "My 1st Great Passphrase!!" For example, you could enter a passphrases such as "We all came out to Montreux" or "My 1st Great Passphrase!!" The security level indicator can help you evaluate the strength of your passphrase: (deep green is the highest level) El indicador de nivel de seguridad puede ayudarle a evaluar la fuerza de tu frase secreta:

(el verde intenso es el nivel más alto)

#### 4. Introduzca para confirmar su frase secreta segura.

A continuación, se inicia el proceso de encriptación. If the MSC was opened in standard mode, the application is reopened in maintenance mode.

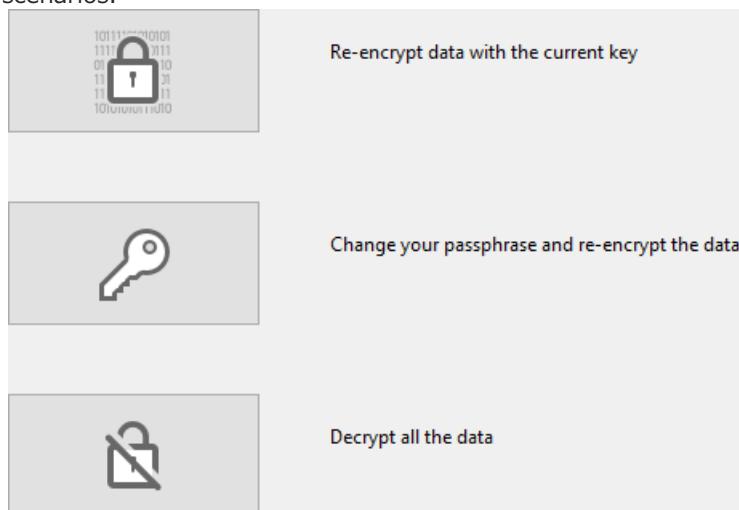
4D ofrece guardar la llave de encriptación (ver [Guardar la llave de encriptación](#) más abajo). Puedes hacerlo en este momento o más adelante. También puedes abrir el archivo de historial de encriptación.

Si el proceso de encriptación es exitoso, la página de encriptación muestra los botones de operaciones de mantenimiento de encriptación.

Atención: durante la operación de encriptación, 4D crea un nuevo archivo de datos vacío y lo llena con los datos del archivo de datos original. Los registros correspondientes a las tablas "encriptadas" se encriptan y luego se copian, los demás registros sólo se copian (también se ejecuta una operación de compactación). Si la operación tiene éxito, el archivo de datos original se traslada a una carpeta de "Archivos reemplazados (encriptados)". If you intend to deliver an encrypted data file, make sure to move/remove any unencrypted data file from the application folder beforehand.

## Operaciones de mantenimiento de la encriptación

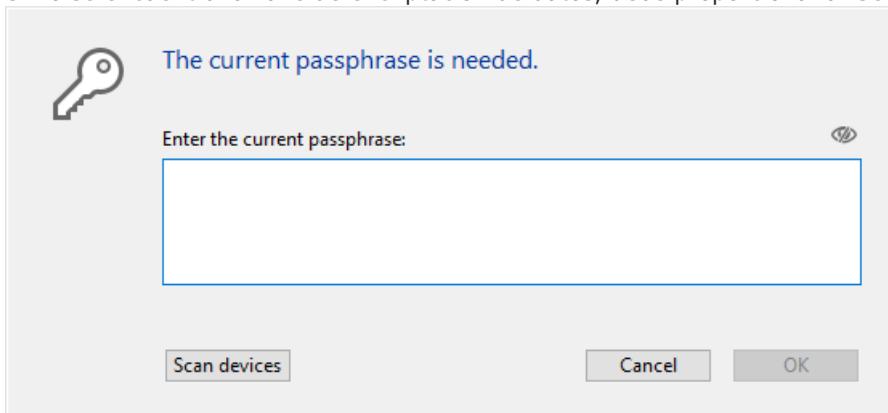
When an application is encrypted (see above), the Encrypt page provides several encryption maintenance operations, corresponding to standard scenarios.



### Suministrar la llave de encriptación de datos actual

Por razones de seguridad, todas las operaciones de mantenimiento del cifrado requieren necesitan la llave de cifrado de datos actual.

- Si la llave de encriptación de datos ya está cargada en el llavero 4D(1), ésta es reutilizada automáticamente por 4D.
- Si no se encuentra la llave de encriptación de datos, debe proporcionarla. Se muestra la siguiente caja de diálogo:



En este paso, tiene dos opciones:

- introduzca la frase secreta actual(2) y haga clic en OK. O
- conecte un dispositivo como una llave USB y haga clic en el botón Escanear dispositivos.

(1) El llavero de 4D almacena todas las llaves de encriptación de datos válidas introducidas durante la sesión de la aplicación.

(2) La frase de contraseña actual corresponde a la contraseña utilizada para generar la llave de cifrado actual.

En todos los casos, si se proporciona información válida, 4D se reinicia en modo de mantenimiento (si no es ya el caso) y ejecuta la operación.

## Volver a cifrar los datos con la llave de cifrado actual

Esta operación es útil cuando se ha modificado el atributo Encriptable de una o varias tablas que contienen los datos. En este caso, para evitar incoherencias en el archivo de datos, 4D impide cualquier acceso de escritura a los registros de las tablas en la aplicación. Entonces es necesario volver a cifrar los datos para restablecer un estado de cifrado válido.

1. Haga clic en Recibir los datos con la llave de cifrado actual.
2. Introduzca la llave de encriptación de datos actual.

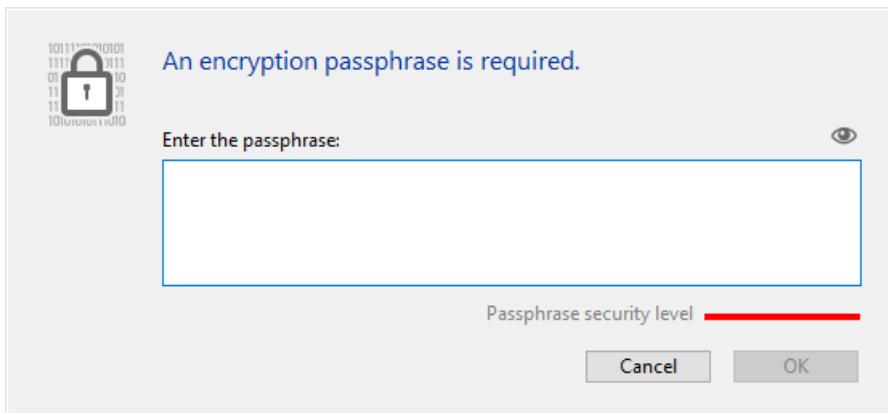
El archivo de datos se vuelve a cifrar correctamente con la llave actual y se muestra un mensaje de confirmación:



## Cambiar la frase secreta y volver a encriptar los datos

Esta operación es útil cuando se necesita cambiar la llave de datos de encriptación actual. Por ejemplo, es posible que tenga que hacerlo para cumplir con las normas de seguridad (como la exigencia de cambiar la frase secreta cada tres meses).

1. Haga clic en Cambiar su frase de contraseña y volver a cifrar los datos.
2. Introduzca la llave de encriptación de datos actual.
3. Enter the new passphrase (for added security, you are prompted to enter it twice):



The data file is encrypted with the

new key and the confirmation message is displayed.



## Desencriptar todos los datos

Esta operación elimina toda la codificación del archivo de datos. Si ya no desea tener sus datos encriptados:

1. Haga clic en Desencriptar todos los datos.
2. Introduzca la llave de encriptación de datos actual ( ver Suministrar la llave de encriptación de datos actual).

El archivo de datos se descifra completamente y se muestra un mensaje de confirmación:



Una vez descifrado el archivo de datos, el estado de cifrado de las tablas no coincide con sus atributos Encriptables. Para restablecer un estado coincidente, debe anular la selección de todos los atributos Encriptable al nivel de la estructura de la base.

## Guardar la llave de encriptación

4D le permite guardar la llave de encriptación de datos en un archivo dedicado. Storing this file on an external device such a USB key will facilitate the use of an encrypted application, since the user would only need to connect the device to provide the key before opening the application in order to access encrypted data.

Puede guardar la llave de encriptación cada vez que se proporcione una nueva frase secreta:

- when the application is encrypted for the first time,
- when the application is re-encrypted with a new passphrase.

Las llaves de encriptación sucesivas pueden ser almacenadas en el mismo dispositivo.

## Archivo de historial

After an encryption operation has been completed, 4D generates a file in the Logs folder of the application. It is created in XML format and named "*ApplicationName\_Encrypt\_Log\_*yyyy-mm-dd hh-mm-ss.xml" or "*ApplicationName\_Decrypt\_Log\_*yyyy-mm-dd hh-mm-ss.xml".

Cada vez que se genera un nuevo archivo de registro, aparece un botón para abrirlo en la página del CSM.

El archivo de historial lista todas las operaciones internas ejecutadas relacionadas con el proceso de cifrado/descifrado, así como los errores (si los hay).

# Copia de seguridad

Una copia de seguridad puede iniciarse de tres maneras:

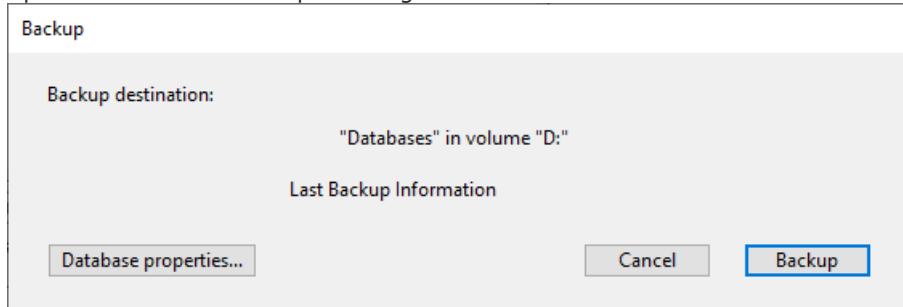
- Manualmente, utilizando el comando Copia de seguridad... del menú 4D Archivo o el botón Copia de seguridad del [Centro de mantenimiento y seguridad](#).
- Automáticamente, utilizando el programador que se puede definir en las [Propiedades](#),
- Por programación, utilizando el comando `BACKUP`.

4D Server: es posible iniciar una copia de seguridad manualmente desde una máquina remota mediante un método que llama al comando `BACKUP`. El comando se ejecutará, en todos los casos, en el servidor.

## Copia de seguridad manual

1. Seleccione el comando Backup... en el menú 4D Archivo.

Aparece la ventana de copia de seguridad:



Puede ver la ubicación de la carpeta de la copia de seguridad mediante el menú emergente situado junto al área "Destino de la copia de seguridad". Esta ubicación se define en la página [Copia de seguridad/configuración de las Propiedades de la base](#).

- También puede abrir el [Centro de mantenimiento y seguridad](#) de 4D y mostrar la [página de copias de seguridad](#).

El botón Propiedades de la base... hace que se muestre la página [Copia de seguridad/Configuración de las Propiedades de la estructura](#).

2. Haga clic en Copia de seguridad para iniciar la copia de seguridad utilizando los parámetros actuales.

## Backup automático periódico

Las copias de seguridad programadas se inician automáticamente. Se configuran en la página [Backup/Planificador de las \\*\\* Propiedades\\*\\*](#).

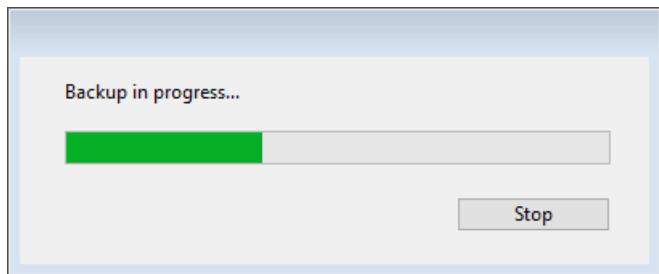
Las copias de seguridad se realizan automáticamente a las horas definidas en esta página sin ningún tipo de intervención del usuario. Para más información sobre el uso de esta caja de diálogo, consulte [Definir las copias de seguridad periódicas](#).

## Comando BACKUP

Puede utilizar los métodos base `On Backup Startup` and `On Backup Shutdown` para controlar el proceso de copia de seguridad (consulte el manual *Lenguaje de 4D*). Puede utilizar el método base `On Backup Startup` y `On Backup Shutdown`, para controlar el proceso de backup (ver el manual *Lenguaje de 4D*).

## Cómo funciona la copia de seguridad

Una vez iniciada la copia de seguridad, 4D muestra una caja de diálogo con un termómetro que indica el progreso de la copia de seguridad:



Este termómetro también se muestra en la página [Backup del CSM](#) si ha utilizado esta caja de diálogo.

El botón Parar permite al usuario interrumpir la copia de seguridad en cualquier momento (consulte [Manejar los problemas de la copia de seguridad](#) más adelante).

El estado de la última copia de seguridad (correcta o fallida) se almacena en el área de información de la [página de copias de seguridad en el CSM](#) o en la página de mantenimiento de 4D Server. También se registra en la base Backup journal.txt.

## Acceso a la aplicación durante la copia de seguridad

4D bloquea los procesos relacionados con los tipos de archivos incluidos en la copia de seguridad: si sólo se hace una copia de seguridad de los archivos del proyecto, no se podrá acceder a la estructura pero sí a los datos. Durante una copia de seguridad, el acceso a la aplicación está restringido por 4D en función del contexto.

Por el contrario, si sólo se hace una copia de seguridad del archivo de datos, se sigue permitiendo el acceso a la estructura. En este caso, las posibilidades de acceso a la aplicación son las siguientes:

- Con 4D versión monopuesto, la aplicación está bloqueada tanto para lectura como para escritura; todos los procesos están congelados. No se puede realizar ninguna acción.
- Con 4D Server, la aplicación sólo está bloqueada en escritura; las máquinas cliente pueden ver los datos. Si una máquina cliente envía una petición de adición, eliminación o cambio al servidor, aparece una ventana que pide al usuario que espere hasta el final de la copia de seguridad. Una vez guardada la aplicación, la ventana desaparece y se realiza la acción. Para cancelar la petición en proceso y no esperar a que finalice la copia de seguridad, basta con hacer clic en el botón Cancelar la operación. Sin embargo, si la acción que espera ser ejecutada proviene de un método lanzado antes de la copia de seguridad, no debe cancelarla porque sólo se cancelan las operaciones que quedan por realizar. Además, un método parcialmente ejecutado puede causar inconsistencias lógicas en los datos.  
➢ Cuando la acción que espera ser ejecutada proviene de un método y el usuario hace clic en el botón Cancelar operación, 4D Server devuelve el error -9976 (Este comando no puede ejecutarse porque la copia de seguridad de la base está en progreso).

## Gestión de los problemas de las copias de seguridad

Puede ocurrir que una copia de seguridad no se ejecute correctamente. Puede haber varias causas de fallo en la copia de seguridad: interrupción del usuario, archivo adjunto no encontrado, problemas en el disco de destino, transacción incompleta, etc. 4D procesa la incidencia según la causa.

En todos los casos, tenga en cuenta que el estado de la última copia de seguridad (correcta o fallida) se almacena en el área de información de la [página de copias de seguridad en el CSM](#) o en la página de mantenimiento de 4D Server, así como en el Backup journal.txt.

- Interrupción del usuario: el botón Parar de la caja de diálogo de progreso permite a los usuarios interrumpir la copia de seguridad en cualquier momento. En este caso, la copia de elementos se detiene y se genera el error 1406. Puedes interceptar este error en el método base `On Backup Shutdown`.
- Archivo adjunto no encontrado: cuando no se encuentra un archivo adjunto, 4D realiza una copia de seguridad parcial (copia de seguridad de los archivos de la aplicación y de los archivos adjuntos accesibles) y devuelve un error.
- Copia de seguridad imposible (disco lleno o protegido contra escritura, disco ausente, fallo de disco, transacción incompleta, aplicación no lanzada en el momento de la copia de seguridad automática programada, etc.): si se trata de un primer error, 4D hará un segundo intento de realizar la copia de seguridad. La espera entre los dos intentos se define en la página Backup/Backup y Restauración de las Propiedades. Si el segundo intento falla, se muestra una caja de diálogo de alerta del sistema y se genera un error. Puedes interceptar este error en el método base `On Backup Shutdown`.

## Historial de copias de seguridad (Backup Journal)

Para facilitar el seguimiento y la verificación de las copias de seguridad, el módulo de copia de seguridad escribe un resumen de cada operación realizada en un archivo especial, que es similar a un diario de actividades. Al igual que un manual de a bordo, todas las operaciones de la base de datos (copias de seguridad, restauraciones, integraciones de archivos de registro) se registran en este archivo, tanto si se han programado como si se han realizado manualmente. La fecha y la hora en que se produjeron estas operaciones también se anotan en el diario.

El historial de copia de seguridad se llama "Backup Journal[001].txt" y se coloca en la carpeta "Logs" del proyecto. El historial de copias de seguridad puede abrirse con cualquier editor de texto.

### Gestión del tamaño del historial de copias de seguridad

En determinadas estrategias de copia de seguridad (por ejemplo, en el caso de que se realicen copias de seguridad de numerosos archivos adjuntos), el historial de copias de seguridad puede alcanzar rápidamente un gran tamaño. Se pueden utilizar dos mecanismos para controlar este tamaño:

- Copia de seguridad automática: antes de cada copia de seguridad, la aplicación examina el tamaño del archivo historial de copia de seguridad actual. Si es superior a 10 MB, se archiva el archivo actual y se crea un nuevo archivo con el número [xxx] incrementado, por ejemplo "Backup Journal[002].txt". Una vez alcanzado el número de archivo 999, la numeración vuelve a empezar por el 1 y los archivos existentes serán sustituidos.
- Posibilidad de reducir la cantidad de información registrada: para ello, basta con modificar el valor de la llave `VerboseMode` en el archivo `Backup.4DSettings` del proyecto. Por defecto, esta llave está definida como True. Si cambia el valor de esta llave a False, sólo se almacenará en el diario de copias de seguridad la información principal: fecha y hora de inicio de la operación y los errores encontrados. Las llaves XML relativas a la configuración de la copia de seguridad se describen en el manual *Backup de las llaves XML 4D*.

## backupHistory.json

Toda la información relativa a las últimas operaciones de copia de seguridad y restauración se almacena en el archivo `backupHistory.json` de la aplicación. Registra la ruta de cada archivo guardado (incluidos los adjuntos), así como el número, la fecha, la hora, la duración y el estado de cada operación. Para limitar el tamaño del archivo, el número de operaciones registradas es el mismo que el número de copias de seguridad disponibles ("Conservar sólo los últimos X archivos de copia de seguridad") definido en las propiedades de la copia de seguridad.

El archivo `backupHistory.json` se crea en la carpeta de destino de la copia de seguridad actual. Puede obtener la ruta real de este archivo utilizando la siguiente declaración:

```
$backupHistory:=Get 4D file(Backup history file)
```

### Atención

La eliminación o el desplazamiento del archivo `backupHistory.json` hará que se reinicie el siguiente número de copia de seguridad. El archivo `backupHistory.json` está formateado para ser utilizado por la aplicación 4D. Si lo que busca es un informe legible en las operaciones de copia de seguridad, quizás le resulte más preciso el diario de copias de seguridad.

# Parámetros de la copia de seguridad

Los parámetros de copia de seguridad se definen a través de tres páginas en el [cuadro de diálogo de los parámetros](#). Puede definir:

- la periodicidad de las copias de seguridad automáticas
- los archivos a incluir en cada copia de seguridad
- las funcionalidades avanzadas permiten ejecutar tareas automáticas

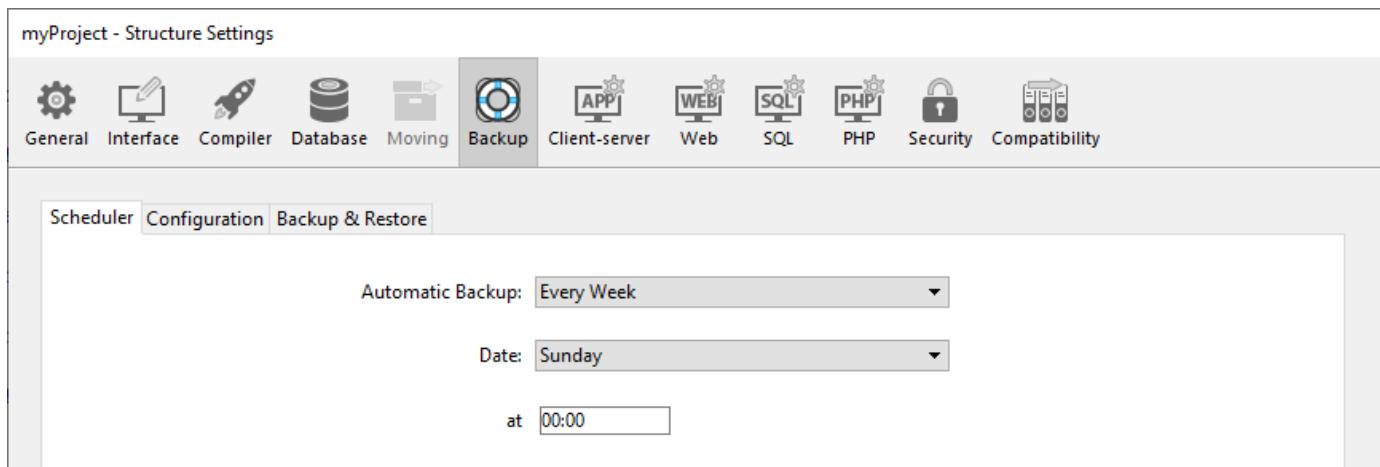
Las propiedades definidas en esta caja de diálogo se escriben en el archivo *Backup.4DSettings*, guardado en la carpeta *Settings*.

## Backups periódicos

Puede automatizar la copia de seguridad de las aplicaciones abiertas con 4D o 4D Server (incluso cuando no hay máquinas cliente conectadas). Esto implica definir una frecuencia de copia de seguridad (en horas, días, semanas o meses); para cada sesión, 4D inicia automáticamente una copia de seguridad utilizando la configuración de copia de seguridad actual.

Si esta aplicación no se inició en el momento teórico de la copia de seguridad, la próxima vez que se inicie 4D, considerará que la copia de seguridad ha fallado y procederá según lo establecido en las Propiedades (consulte [Manejo de problemas de la copia de seguridad](#)).

Los parámetros de copia de seguridad periódicos se definen en la página [Backup/Periodicidad de las Propiedades](#):



Las opciones que se encuentran en esta pestaña le permiten establecer y configurar las copias de seguridad periódicas automáticas de la aplicación. Puede elegir una configuración rápida estándar o puede personalizarla completamente. Aparecen varias opciones en función de la elección realizada en el menú Copia de seguridad automática :

- Nunca: la función de copia de seguridad programada está inactiva.
- Cada hora: programa una copia de seguridad automática cada hora, a partir de la hora siguiente.
- Cada día: programa una copia de seguridad automática cada día. A continuación, puede introducir la hora a la que debe comenzar la copia de seguridad.
- Todas las semanas: programa una copia de seguridad automática cada semana. Dos áreas de entrada adicionales le permiten indicar el día y la hora en que debe comenzar la copia de seguridad.
- Todos los meses: programa una copia de seguridad automática cada mes. Dos áreas de entrada adicionales le permiten indicar el día del mes y la hora en que debe comenzar la copia de seguridad.
- Personalizado: sirve para configurar las copias de seguridad automáticas "a medida". Al seleccionar esta opción, aparecen varias áreas de entrada adicionales:
  - Cada X hora(s): permite programar las copias de seguridad con una base horaria. Puede introducir un valor entre 1 y 24.
  - Cada X día(s) a las x: permite programar las copias de seguridad con una base diaria. Por ejemplo, introduzca 1 si desea realizar una copia de seguridad diaria. Cuando esta opción está marcada, debe introducir la hora a la

que debe comenzar la copia de seguridad.

- Cada X semana(s) a las x: permite programar las copias de seguridad con una base semanal. Introduzca 1 si desea realizar una copia de seguridad semanal. Cuando esta opción está marcada, debe introducir el día(s) de la semana y la hora que debe comenzar la copia de seguridad. Si lo desea, puede seleccionar varios días de la semana. Por ejemplo, puede utilizar esta opción para definir dos copias de seguridad semanales: una el miércoles y otra el viernes.
- Cada X mes(es), X día a x: Permite programar copias de seguridad de forma mensual. Introduzca 1 si desea realizar una copia de seguridad mensual. Cuando esta opción está marcada, debe indicar el día de cada mes y la hora a la cual debe comenzar la copia de seguridad.

Los cambios de la hora estándar a la hora de verano podrían afectar temporalmente al programador automático y activar la siguiente copia de seguridad con un cambio de hora de una hora. Esto ocurre sólo una vez y las siguientes copias de seguridad se ejecutan a la hora prevista.

## Configuración

La página Copia de seguridad/Configuración de las Propiedades permite designar los archivos de copia de seguridad y su ubicación, así como la del archivo de historial. Estos parámetros son específicos de cada aplicación abierta por 4D o 4D Server.

myProject - Structure Settings

General Interface Compiler Database Moving Backup Client-server Web SQL PHP Security Compatibility

Scheduler Configuration Backup & Restore

Content

Data  
 Structure  
 User Structure (only for binary database)

Attachments:

./Data/data.4DIndx  
./Data/data.4DSyncData  
./Data/data.4DSyncHeader

Delete Add folder... Add file...

Backup File Destination Folder

"myProject" in volume "D:"

Used Space: 130,19 Go Free Space: 346,26 Go

Log Management

Use Log:  
"data.journal" in volume "D:"

4D Server: estos parámetros sólo se pueden configurar desde la máquina 4D Server.

## Contenido

Esta área le permite determinar qué archivos y/o carpetas deben copiarse durante la siguiente copia de seguridad.

- Datos: archivo de datos de la aplicación. Cuando esta opción está marcada, los siguientes elementos se copian

automáticamente al mismo tiempo que los datos:

- el archivo de historial actual de la aplicación (si existe),
  - la carpeta `Settings` completa situada [junto al archivo de datos](#) (si existe), es decir, *los parámetros usuario para los datos*.
- Estructura: carpetas y archivos proyecto de la aplicación. En el caso de proyectos compilados, esta opción permite hacer una copia de seguridad del archivo `.4dz`. Cuando esta opción está marcada, se hace una copia de seguridad automática de la carpeta completa `Settings` situada [en el mismo nivel que la carpeta Project](#), es decir, los *parámetros usuario*.
  - Archivo de estructura usuario (sólo para bases binaria) : *funcionalidad obsoleta*
  - Adjuntos: esta área permite especificar un conjunto de archivos y/o carpetas que se respaldarán al mismo tiempo que la aplicación. Estos archivos pueden ser de cualquier tipo (documentos o plantillas de plug-ins, etiquetas, informes, imágenes, etc.). Puede definir archivos individuales o carpetas cuyo contenido se respaldará completamente. Cada elemento adjunto aparece con su ruta de acceso completa en el área "Adjuntos".
    - Eliminar: retira el archivo seleccionado de la lista de archivos adjuntos.
    - Añadir carpeta...: muestra una caja de diálogo que permite seleccionar una carpeta para añadirla a la copia de seguridad. En el caso de una restitución, la carpeta se recuperará con su estructura interna. Puede seleccionar toda carpeta o volumen conectado a la máquina, a excepción de la carpeta que contiene los archivos de la aplicación.
    - Añadir archivo...: muestra una caja de diálogo que permite seleccionar un archivo para añadirlo a la copia de seguridad.

## Carpeta de destino del archivo de copia de seguridad

Esta área le permite visualizar y cambiar la ubicación en la que se almacenarán los archivos de copia de seguridad, así como los archivos de copia de seguridad del archivo historial (si procede).

Para ver la ubicación de los archivos, haga clic en el área para que aparezca su ruta de acceso en un menú emergente.

Para modificar la ubicación donde se almacenan estos archivos, haga clic en el botón .... Aparece una caja de diálogo de selección, que permite seleccionar la carpeta o el disco donde se colocarán las copias de seguridad. Las áreas "Espacio utilizado" y "Espacio libre" se actualizan automáticamente e indican el espacio restante en el disco de la carpeta seleccionada.

## Gestión del archivo de historial

La opción Utilizar el archivo de historial, cuando está marcada, indica que la aplicación utiliza un archivo de historial. Su ruta de acceso se especifica debajo de la opción. Cuando esta opción está marcada, no es posible abrir la aplicación sin un archivo de historial.

Por defecto, todo proyecto creado con 4D utiliza un archivo de historial (opción Utilizar archivo de historial seleccionada en la página General de las Preferencias). El archivo de historial se llama `data.journal` y se coloca en la carpeta Data.

La activación de un nuevo archivo de historial requiere una copia de seguridad previa de los datos de la aplicación. Al marcar esta opción, un mensaje de advertencia le informa de que es necesario realizar una copia de seguridad. La creación del archivo de historial se pospone y se creará realmente sólo después de la siguiente copia de seguridad de la aplicación.

## Copia de seguridad y restitución

La modificación de las opciones de copia de seguridad y restauración es opcional. Sus valores por defecto corresponden a un uso estándar de la función.

myProject - Structure Settings

## Parámetros generales

- Conservar únicamente los últimos X archivos de copia de seguridad : este parámetro activa y configura el mecanismo utilizado para eliminar los archivos de copia de seguridad más antiguos, lo que evita el riesgo de saturar la unidad de disco. Esta funcionalidad opera de la siguiente manera: una vez finalizado el backup actual, 4D elimina el archivo más antiguo si se encuentra en la misma ubicación que el archivo del que se está haciendo el backup y tiene el mismo nombre (puede solicitar que el archivo más antiguo se elimine antes del backup para ahorrar espacio). Si, por ejemplo, el número de conjuntos se define en 3, las tres primeras copias de seguridad crean los archivos MyBase-0001, MyBase-0002 y MyBase-0003 respectivamente. Durante la cuarta copia de seguridad, se crea el archivo MyBase-0004 y se elimina MyBase-0001. Por defecto, el mecanismo de eliminación de conjuntos está activado y 4D guarda 3 conjuntos de copia de seguridad. Para desactivar el mecanismo, basta con deseleccionar la opción.

Este parámetro se refiere tanto a las copias de seguridad de la aplicación como de los archivos del historial.

- Copia de seguridad sólo si el archivo de datos ha sido modificado : cuando se marca esta opción, 4D inicia las copias de seguridad programadas sólo si se han añadido, modificado o eliminado datos desde la última copia de seguridad. De lo contrario, la copia de seguridad programada se cancela y se pospone hasta la siguiente copia de seguridad programada. No se genera ningún error; sin embargo, el diario de copias de seguridad señala que la copia de seguridad se ha pospuesto. Esta opción también permite ahorrar tiempo de máquina para la copia de seguridad de las aplicaciones utilizadas principalmente para su visualización. Tenga en cuenta que al activar esta opción no se tienen en cuenta las modificaciones realizadas en los archivos de estructura o en los archivos adjuntos.

Este parámetro se refiere tanto a las copias de seguridad de la aplicación como de los archivos del historial.

- Eliminar el archivo de copia de seguridad más antiguo antes/después de la copia de seguridad : esta opción sólo se utiliza si la opción "Conservar sólo los últimos X archivos de copia de seguridad" está marcada. Especifica si 4D debe comenzar borrando el archivo más antiguo antes de iniciar la copia de seguridad (antes opción) o si el borrado

debe tener lugar una vez finalizada la copia de seguridad (opción después). Para que este mecanismo funcione, el archivo más antiguo no debe haber sido renombrado o movido.

- Si falla la copia de seguridad : esta opción permite configurar el mecanismo utilizado para gestionar las copias de seguridad fallidas (copia de seguridad imposible). Cuando no se puede realizar una copia de seguridad, 4D le permite realizar un nuevo intento.
  - Reintentar en la siguiente fecha y hora programada : esta opción sólo tiene sentido cuando se trabaja con copias de seguridad automáticas programadas. Equivale a anular la copia de seguridad fallida. Se genera un error.
  - Reintentar después de X segundo(s), minuto(s) u hora(s) : cuando se marca esta opción, se ejecuta un nuevo intento de copia de seguridad después del periodo de espera. Este mecanismo permite anticipar ciertas circunstancias que pueden bloquear la copia de seguridad. Puede establecer un periodo de espera en segundos, minutos u horas utilizando el menú correspondiente. Si el nuevo intento también falla, se genera un error y se anota el fallo en el área de estado de la última copia de seguridad y en el archivo del diario de copias de seguridad.
  - Cancelar la operación después de X intentos : este parámetro se utiliza para definir el número máximo de intentos de copia de seguridad fallidos. Si la copia de seguridad no se ha realizado con éxito una vez alcanzado el número máximo de intentos establecido, se cancela y se genera el error 1401 ("Se ha alcanzado el número máximo de intentos de copia de seguridad; la copia de seguridad automática está temporalmente desactivada"). En este caso, no se intentará realizar una nueva copia de seguridad automática mientras no se haya reiniciado la aplicación o se haya realizado con éxito una copia de seguridad manual. Este parámetro es útil para evitar un caso en el que un problema prolongado (que requiera la intervención humana) que impidiera la realización de una copia de seguridad hubiera llevado a la aplicación a intentar repetidamente la copia de seguridad en detrimento de su rendimiento general. Por defecto, este parámetro no está seleccionado.

4D considera que una copia de seguridad ha fallado si la aplicación no se ha iniciado en el momento en que se ha programado la realización de la copia de seguridad automática.

## Archivo

Estas opciones se aplican a los archivos de copia de seguridad principales y a los archivos de copia de seguridad del historial.

- Tamaño del segmento (Mb) 4D le permite segmentar los archivos, es decir, cortarlos en tamaños más pequeños. Este funcionamiento permite, por ejemplo, almacenar una copia de seguridad en varios discos diferentes (DVD, dispositivos usb, etc.). Durante la restauración, 4D fusionará automáticamente los segmentos. Cada segmento se llama MyApplication[xxxx-yyyy].4BK, donde xxxx es el número de copia de seguridad e yyyy es el número de segmento. Por ejemplo, los tres segmentos de la copia de seguridad de la base MyApplication se llaman MyApplication[0006-0001].4BK, MyApplication[0006-0002].4BK y MyApplication[0006-0003].4BK. El menú Tamaño del segmento es un combo box que permite establecer el tamaño en MB de cada segmento de la copia de seguridad. Puede elegir uno de los tamaños preestablecidos o introducir un tamaño específico entre 0 y 2048. Si se pasa 0, no se produce ninguna segmentación (esto equivale a pasar Ninguna).
- Tasa de compresión Por defecto, 4D comprime las copias de seguridad para ayudar a ahorrar espacio en el disco. Sin embargo, la fase de compresión de archivos puede ralentizar notablemente las copias de seguridad cuando se trata de grandes volúmenes de datos. La opción Tasa de compresión permite ajustar la compresión de los archivos:
  - Ninguna: no se aplica ninguna compresión de archivos. La copia de seguridad es más rápida, pero los archivos son considerablemente más grandes.
  - Rápido (por defecto): esta opción es un compromiso entre la velocidad de la copia de seguridad y el tamaño del archivo.
  - Compactado: la tasa de compresión máxima se aplica a los archivos. Los ficheros de archivos ocupan el menor espacio posible en el disco, pero la copia de seguridad se ralentiza notablemente.
- Tasa de entrelazamiento y tasa de redundancia 4D genera archivos mediante algoritmos específicos que se basan en mecanismos de optimización (entrelazamiento) y seguridad (redundancia). Puedes configurar estos mecanismos en función de sus necesidades. Los menús asociados a estas opciones contienen índices de Bajo, Medio, Alto y Ninguno (por defecto).
  - Tasa de entrelazamiento: el entrelazamiento consiste en almacenar los datos en sectores no adyacentes para limitar los riesgos en caso de daño de los sectores. Cuanto mayor sea la tasa, mayor será la seguridad; sin embargo, el procesamiento de datos utilizará más memoria.

- Tasa de redundancia: la redundancia permite asegurar los datos presentes en un fichero repitiendo varias veces la misma información. Cuanto mayor sea la tasa de redundancia, mayor será la seguridad de los archivos; sin embargo, el almacenamiento será más lento y el tamaño de los archivos aumentará en consecuencia.

## Restauración automática

- Restaurar la última copia de seguridad si la base está dañada : cuando se marca esta opción, el programa inicia automáticamente la restauración del archivo de datos de la última copia de seguridad válida de la aplicación, si se detecta una anomalía (archivo dañado, por ejemplo) durante el lanzamiento de la aplicación. No se requiere ninguna intervención por parte del usuario; sin embargo, la operación se registra en el diario de copias de seguridad.
- Integrar el último archivo de historial si la base está incompleta : cuando se marca esta opción, el programa integra automáticamente el archivo de historial al abrir o restaurar la aplicación.
  - Durante la apertura de una aplicación, el archivo de historial actual se integra automáticamente si 4D detecta que hay operaciones almacenadas en el archivo de historial que no están presentes en los datos. Esta situación se produce, por ejemplo, si se produce un corte de energía cuando hay operaciones en la caché de datos que aún no se han escrito en el disco.
  - Al restaurar la aplicación, si el archivo de historial actual o un archivo de copia de seguridad del historial que tiene el mismo número que el archivo de copia de seguridad se almacena en la misma carpeta, 4D examina su contenido. Si contiene operaciones que no se encuentran en el archivo de datos, el programa las integra automáticamente.

El usuario no ve ninguna caja de diálogo; la operación es completamente automática. El objetivo es facilitar al máximo su uso. La operación se registra en el diario de copias de seguridad.

En el caso de una restauración automática, sólo se restauran los siguientes elementos: - Archivo .4DD - archivo .4DIndx - Archivo .4DSyncData - Archivo .4DSyncHeader - Carpeta de datos externos

Si desea obtener los archivos adjuntos o los archivos del proyecto, debe realizar una [restauración manual](#).

# Archivo de historial (.journal)

Una aplicación de uso continuo siempre está registrando cambios, adiciones o supresiones. Realizar copias de seguridad periódicas de los datos es importante, pero no permite (en caso de incidente) restaurar los datos introducidos desde la última copia de seguridad. Para responder a esta necesidad, 4D ofrece ahora una herramienta específica: el archivo de historial. Este archivo permite garantizar la seguridad permanente de los datos.

Además, 4D trabaja continuamente con una caché de datos en la memoria. Esto acelera el funcionamiento de las aplicaciones; de hecho, el acceso a la memoria es más rápido que el acceso al disco duro. Todos los cambios realizados en los datos de la aplicación se almacenan temporalmente en la caché antes de escribirse en el disco duro. Si se produce un incidente en la aplicación antes de que los datos almacenados en la caché puedan escribirse en el disco, deberá incluir el archivo de historial actual para poder recuperar la aplicación por completo.

Por último, 4D dispone de funciones que analizan el contenido del archivo de historial, permitiendo revertir las operaciones realizadas sobre los datos de la aplicación. Estas funciones están disponibles en el CSM: consulte la página [Análisis de actividades](#) y la página [Retroceder](#).

## Funcionamiento del archivo de historial

El archivo de historial generado por 4D contiene una descripción de todas las operaciones realizadas en los datos de las tablas registradas en el diario, que se registran de forma secuencial. Por defecto, todas las tablas se registran en el diario, es decir, se incluyen en el archivo de historial, pero puede anular la selección de tablas individuales mediante la propiedad **Incluir** en el archivo de historial.

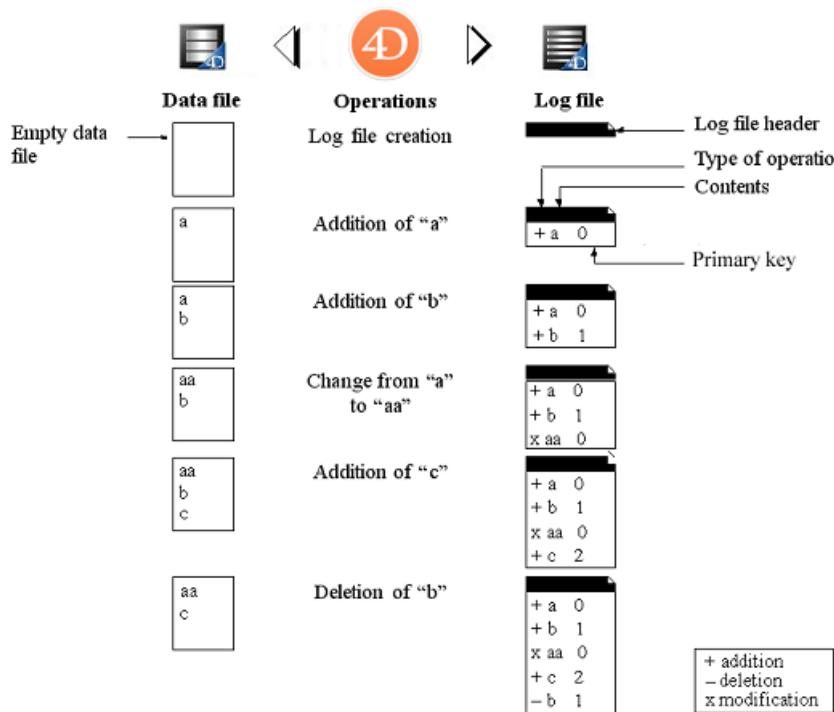
Así, cada operación realizada por un usuario provoca dos acciones simultáneas: la primera en el archivo de datos (la instrucción se ejecuta normalmente) y la segunda en el archivo de historial (se registra la descripción de la operación). El archivo de historial se crea de forma independiente, sin perturbar ni ralentizar el trabajo del usuario. Una aplicación sólo puede trabajar con un archivo de historial a la vez. El archivo de historial registra los siguientes tipos de acciones:

- Apertura y cierre del archivo de datos,
- Apertura y cierre del proceso (contextos),
- Adición de registros o BLOBs,
- Modificación de registros,
- Eliminación de registros,
- Creación y cierre de transacciones.

Para más información sobre estas acciones, consulte la página [Análisis de actividades](#) del CSM.

4D gestiona el archivo de historial. Tiene en cuenta todas las operaciones que afectan al archivo de datos por igual, independientemente de las manipulaciones realizadas por un usuario, métodos 4D, el motor SQL, los plug-ins, o un navegador web o una aplicación móvil.

La siguiente ilustración resume el funcionamiento del archivo de historial:



El archivo de historial actual se guarda automáticamente con el archivo de datos actual. Este mecanismo tiene dos ventajas distintas:

- Evitar la saturación del volumen de disco donde se almacena el archivo de historial. Sin una copia de seguridad, el archivo de historial se haría cada vez más grande con el uso, y acabaría utilizando todo el espacio disponible en el disco. Para cada copia de seguridad del archivo de datos, 4D o 4D Server cierra el archivo de historial actual e inmediatamente inicia un nuevo archivo vacío, evitando así el riesgo de saturación. A continuación, el archivo de historial antiguo se archiva y, finalmente, se destruye en función del mecanismo de gestión de los conjuntos de copias de seguridad.
- Conservar los archivos de historial correspondientes a las copias de seguridad para poder analizar o reparar una aplicación en un momento posterior. La integración de un archivo de historial sólo puede hacerse en la aplicación a la que corresponde. Para poder integrar correctamente un archivo de historial en una copia de seguridad, es importante que las copias de seguridad y los archivos de historial se archiven simultáneamente.

## Crear el archivo de historial

Por defecto, toda aplicación creada con 4D utiliza un archivo de historial (opción definida en la página General de las Preferencias). El archivo de historial se llama *data.journal* y se coloca en la carpeta Data.

Puede averiguar si su aplicación utiliza un archivo de historial en cualquier momento: sólo tiene que comprobar si la opción Utilizar el archivo de historial está seleccionada en la página Backup/Configuración de las Propiedades. Si deselecciona esta opción, o si utiliza una aplicación sin archivo de historial y desea configurar una estrategia de copia de seguridad con un archivo de historial, tendrá que crear uno.

Para crear un archivo de historial:

1. En la página Copia de seguridad/Configuración de las Propiedades de estructura, marque la opción Utilizar el archivo de historial. El programa muestra una caja de diálogo estándar de abrir/nuevo archivo. Por defecto, el archivo de historial se llama *data.journal*.
2. Mantenga el nombre por defecto o cambie el nombre, y luego seleccione la ubicación del archivo. Si tiene al menos dos discos duros, se recomienda colocar el archivo de historial en un disco distinto al que contiene el proyecto de aplicación. Si se pierde el disco duro de la aplicación, aún puede recuperar su archivo de historial.
3. Haga clic en Guardar. El disco y el nombre del archivo de historial abierto se muestran ahora en el área Utilizar historial de la caja de diálogo. Puede hacer clic en esta área para que aparezca un menú emergente con la ruta del historial en el disco.
4. Valide la caja de diálogo de las Propiedades.

Para poder crear directamente un archivo de historial, los datos deben estar en una de las siguientes situaciones:

- El archivo de datos está en blanco,
- Acaba de realizar una copia de seguridad y aún no se han realizado cambios en los datos.

Si hace clic en Aceptar, la copia de seguridad comienza inmediatamente, y luego se activa el archivo de historial. Si hace clic en Aceptar, la copia de seguridad comienza inmediatamente, y luego se activa el archivo de historial. Si hace clic en Cancelar, la solicitud se guarda pero la creación del archivo de historial se pospone y en realidad sólo se creará después de la siguiente copia de seguridad de la aplicación. Esta precaución es indispensable porque, para restaurar una aplicación después de algún incidente, necesitará una copia de la aplicación en la que se integrarán las operaciones registradas en el archivo de historial.

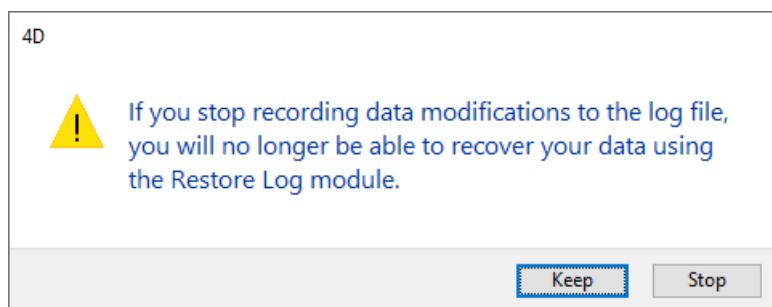
Sin tener que hacer nada más, todas las operaciones realizadas sobre los datos se registran en este archivo y se utilizarán en el futuro cuando se abra la aplicación.

Debe crear otro archivo de historial si crea un nuevo archivo de datos. Debe establecer o crear otro archivo de historial si abre otro archivo de datos que no está asociado a un archivo de historial (o si falta el archivo de historial).

## Cerrar el historial

Si desea dejar de registrar las operaciones en el archivo de historial actual, sólo tiene que anular la selección de la opción Utilizar el archivo de historial en la página Copia de seguridad/Configuración de las Propiedades.

4D muestra entonces un mensaje de alerta para recordarle que esta acción le impide aprovechar la seguridad que ofrece el archivo de historial:



Si hace clic en el botón Parar, el archivo de historial actual se cierra inmediatamente (la caja de diálogo de las Propiedades no necesita ser validada después).

Si desea cerrar el archivo de historial actual porque es demasiado grande, puede considerar la posibilidad de realizar una copia de seguridad del archivo de datos, lo que hará que el archivo de historial se copie también.

**4D Server:** el comando `New log file` cierra automáticamente el archivo de historial actual e inicia uno nuevo. Si por alguna razón el archivo de historial no está disponible durante una sesión de trabajo, se genera el error 1274 y 4D Server no permite a los usuarios escribir más datos. Cuando el archivo de historial está disponible de nuevo, es necesario hacer una copia de seguridad.

# Restaurar

4D le permite restaurar conjuntos enteros de datos de una aplicación en caso de que se presente un incidente, independientemente de sus causas. Pueden producirse dos categorías principales de incidentes:

- La parada inesperada de la aplicación mientras está en uso. Este incidente puede producirse por un corte de luz, un fallo de un elemento del sistema, etc. En este caso, dependiendo del estado actual de la caché de datos en el momento del incidente, la restauración de la aplicación puede requerir diferentes operaciones:
  - Si la caché estaba vacía, la aplicación se abre normalmente. Se registraron todos los cambios realizados en la aplicación. Este caso no requiere ninguna operación particular.
  - Si la caché contiene operaciones, el archivo de datos está intacto pero requiere integrar el archivo de historial actual.
  - Si la caché estaba en proceso de escritura, es probable que el archivo de datos esté dañado. Hay que restaurar la última copia de seguridad e integrar el archivo de historial actual.
- La pérdida de archivo(s) de la aplicación. Este incidente puede producirse por sectores defectuosos en el disco que contiene la aplicación, un virus, un error de manipulación, etc. Hay que restaurar la última copia de seguridad y luego integrar el archivo de historial actual. Para saber si una aplicación ha sido dañada tras un incidente, basta con relanzarla con 4D. El programa realiza un auto diagnóstico y detalla las operaciones de reparación a realizar. En modo automático, estas operaciones se realizan directamente sin ninguna intervención por parte del usuario. Si se ha puesto en marcha una estrategia regular de copias de seguridad, las herramientas de restauración de 4D le permitirán (en la mayoría de los casos) recuperar la aplicación en el estado exacto en que se encontraba antes del incidente.

4D puede lanzar procedimientos automáticamente de recuperación de las aplicaciones tras los incidentes. Estos mecanismos se gestionan mediante dos opciones disponibles en la página Backup/Backup y Restauración de las Propiedades. Para más información, consulte el párrafo [Restauración automática](#).

Si la incidencia es el resultado de una operación inadecuada realizada sobre los datos (eliminación de un registro, por ejemplo), puede intentar reparar el archivo de datos utilizando la función "rollback" del archivo de historial. Esta función está disponible en la página [Retroceder](#) del CSM.

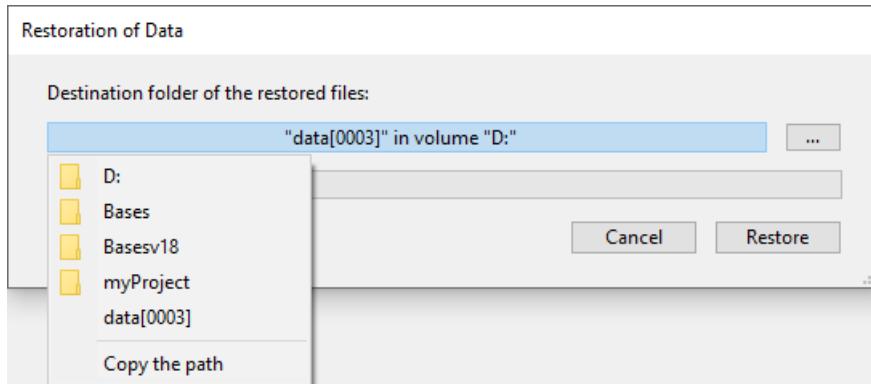
## Restitución manual de una copia de seguridad (diálogo estándar)

Puede restaurar manualmente el contenido de un archivo generado por el módulo de copia de seguridad. Una restauración manual puede ser necesaria, por ejemplo, para restaurar todo el contenido de un archivo (archivos de proyecto y archivos adjuntos), o para realizar búsquedas entre los archivos. La restauración manual también puede realizarse junto con la integración del archivo de registro actual.

La restauración manual de las copias de seguridad puede realizarse a través de la caja de diálogo estándar de apertura de documento, o a través de la página [Restitución](#) del CSM. La restitución a través del CSM ofrece más opciones y permite previsualizar el contenido del archivo. Por otro lado, sólo se pueden restaurar los archivos asociados a la aplicación abierta.

Para restaurar manualmente una aplicación a través de una caja de diálogo estándar:

1. Elija Restituir... en el menú de la aplicación 4D Archivo. It is not mandatory that an application project be open. O Ejecute el comando RESTORE desde un método 4D. Aparece una caja de diálogo estándar de apertura de archivos.
2. Seleccione un archivo de copia de seguridad (.4bk) o un archivo de copia de seguridad del historial (.4bl) que deseé restaurar y haga clic en Abrir. Aparece una caja de diálogo que permite especificar la ubicación donde se restaurarán los archivos. Por defecto, 4D restaura los archivos en una carpeta llamada *Nomarchivo* (sin extensión) situada junto al archivo. Puede mostrar la ruta de acceso:



También puede hacer clic en el botón [...] para especificar una ubicación diferente.

3. Haga clic en el botón Restaurar. 4D extrae todos los archivos de copia de seguridad de la ubicación especificada. Si el archivo de historial actual o un archivo de copia de seguridad del historial tiene el mismo número que el archivo de copia de seguridad se almacena en la misma carpeta, 4D examina su contenido. Si contiene operaciones que no están presentes en el archivo de datos, el programa le pregunta si desea integrar estas operaciones. La integración se realiza automáticamente si la opción de integración automática del historial está seleccionada (ver [Restauración automática](#)).

4.(Opcional) Haga clic en OK para integrar el archivo de historial en la aplicación restaurada. Si la restauración y la integración se han realizado correctamente, 4D muestra una caja de diálogo que indica que la operación se ha realizado con éxito.

5. Haga clic en OK.

Se muestra la carpeta de destino. Durante la restauración, 4D coloca todos los archivos de la copia de seguridad en esta carpeta, independientemente de la posición de los archivos originales en el disco cuando se inicia la copia de seguridad. De esta manera, encontrará sus archivos con mayor facilidad.

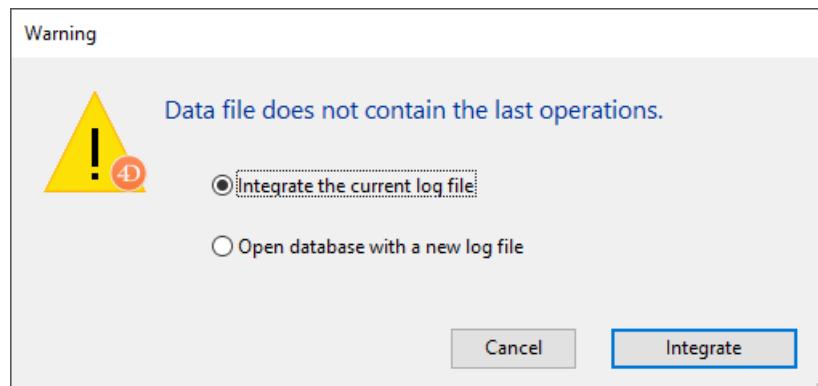
Todo el contenido relacionado con el archivo de datos (carpeta archivos y [Settings](#)) se restaura automáticamente en una subcarpeta [Data](#) dentro de la carpeta de destino.

## Restaurar manualmente una copia de seguridad (CSM)

Puede restaurar manualmente un archivo de la aplicación actual utilizando la página [Restauración](#) del Centro de Mantenimiento y Seguridad (CMS).

## Integración manual del historial

Si no ha marcado la opción de integración automática del archivo de historial en la página Restaurar del CSM (ver [Integración sucesiva de varios archivos de historial](#)), aparece una caja de diálogo de advertencia durante la apertura de la aplicación cuando 4D advierte que el archivo de historial contiene más operaciones de las que se han realizado en el archivo de datos.



Para que este mecanismo funcione, 4D debe poder acceder al archivo de historial en su ubicación actual.

Puede elegir si integrar o no el archivo de historial actual. No integrar el archivo de historial actual permite evitar la reproducción de los errores cometidos en los datos.

# Generalidades

4D en modo local y remoto y 4D Server incluyen un motor de servidor web (también conocido como servidor http) que le permite diseñar y publicar poderosas aplicaciones web que pueden aprovechar al máximo sus bases de datos 4D.

## Fácil de supervisar

Puede iniciar o detener la publicación de la aplicación web en cualquier momento. Para ello, basta con seleccionar un comando del menú o ejecutar una sola línea de código.

Supervisar el servidor web 4D es fácil y se puede hacer utilizando la ventana de administración de 4D Server o a través de [URLs especiales](#).

## Listo para usar

El servidor web 4D crea automáticamente una carpeta raíz y una página de inicio por defecto, disponibles inmediatamente.

## Seguridad

La seguridad de los datos está presente en todas las etapas de las implementaciones del servidor web 4D. Los niveles de seguridad son escalables y la configuración por defecto suele seleccionar las opciones más seguras. La seguridad del servidor web 4D se basa en los elementos siguientes:

- Soporte extendido del [protocolo TLS \(HTTPS\)](#),
- Autenticación: [funciones de autenticación](#) flexibles y personalizables basadas en la configuración integrada, así como también métodos base de reserva ([On Web Authentication](#) para el servidor web y [On REST Authentication](#) para el servidor REST),
- Control de los contenidos expuestos : sólo los elementos que exponga explícitamente pueden estar disponibles desde peticiones web directas o peticiones REST. Debe declarar:
  - [Los métodos proyecto](#) expuestos a través de peticiones HTTP
  - [Las funciones ORDA](#) expuestas a través de peticiones REST
  - [Tablas y campos](#) que no quiere que estén disponibles para las peticiones REST.
- Sandboxing mediante la definición de una [carpeta HTML raíz](#) por defecto,
- Control del uso de los recursos del servidor (por ejemplo, vía la opción [máximo de procesos web concurrentes](#)).

Para una visión general de las funciones de seguridad de 4D, consulte la [Guía de seguridad de 4D](#).

## Sesiones usuario

El servidor web 4D incluye completas funcionalidades automáticas para gestionar fácilmente las [sesiones web](#) (sesiones de usuario) basadas en cookies.

## Punto de acceso para las peticiones REST

El servidor web 4D permite acceder a los datos almacenados en sus aplicaciones 4D a través de peticiones REST. Las peticiones REST ofrecen acceso directo a cualquier operación de la base de datos, como añadir, leer, editar, ordenar o buscar datos.

Las peticiones REST se detallan en la sección [Servidor REST](#).

## Extensión de los parámetros

La configuración del servidor web 4D se define a través de un amplio conjunto de ajustes a nivel de aplicación que también pueden personalizarse para la sesión utilizando las propiedades del objeto `webServer` o el comando `WEB SET OPTION`.

## Plantillas y URLs

El servidor web 4D soporta el acceso a los datos almacenados en sus aplicaciones 4D a través de páginas de plantillas y URLs específicas.

- Las páginas de plantillas contienen [etiquetas especiales](#) que inician el procesamiento del servidor web en el momento en que se envían a los navegadores.
- [Las URLs específicas](#) permiten llamar a 4D para ejecutar cualquier acción; estas URLs también pueden utilizarse como acciones de formulario para activar el procesamiento cuando el usuario publica formularios HTML.

## Métodos base dedicados

[On Web Authentication](#), [On Web Connection](#), así como también los métodos base [On REST Authentication](#) son los puntos de entrada de las peticiones en el servidor web; se pueden utilizar para evaluar y enrutar todo tipo de petición.

# Configuración

Los parámetros del servidor web 4D incluye parámetros de seguridad, puertos de escucha, rutas por defecto y varias opciones que cubren todas las funcionalidades del servidor. 4D ofrece valores por defecto para todos los parámetros.

## ¿Dónde configurar los parámetros?

Hay diferentes maneras de configurar los parámetros del servidor web 4D, en función del alcance y del servidor que se quiera configurar:

Ubicación del parámetro	Alcance	Servidor web a utilizar
<code>objeto webServer</code>	Temporal (sesión actual)	Todos los servidores web, incluidos los servidores web de componentes
<code>WEB SET OPTION</code> o comando <code>WEB XXX</code>	Temporal (sesión actual)	Servidor principal
<code>Settings dialog box</code> (Web pages)	Permanente (todas las sesiones, almacenadas en el disco)	Servidor principal

Algunos parámetros no están disponibles desde todos los lugares.

## Caché

Puede ajustarse con	Nombre	Comentarios
Caja de diálogo de parámetros	<a href="#">Página de configuración/Utilización de la caché Web 4D</a>	
Caja de diálogo de parámetros	<a href="#">Página de configuración/Tamaño de la caché de las páginas</a>	

Activa y configura la caché de las páginas web.

El servidor web 4D dispone de una caché que permite cargar las páginas estáticas, las imágenes GIF, las imágenes JPEG (<512 kb) y las hojas de estilo (archivos.css) en memoria, a medida que se solicitan. El uso de la caché permite aumentar considerablemente el rendimiento del servidor web cuando se envían páginas estáticas. El caché se comparte entre todos los procesos web. When the cache is enabled, the 4D Web server looks for any static page requested by the browser in the cache first. Si encuentra la página, la envía inmediatamente. If not, 4D loads the page from disk and places it in the cache.

Puede modificar el tamaño de la caché en el área Tamaño de la caché de las páginas . El valor a definir depende del número y del tamaño de las páginas estáticas de su sitio web, así como de los recursos de que dispongan las máquinas locales.

Mientras utiliza su base de datos web, puede verificar el rendimiento de la caché utilizando el comando `WEB GET STATISTICS` . Si, por ejemplo, observa que la tasa de uso de la caché se acerca al 100%, puede considerar aumentar el tamaño que se le ha asignado. Los URL [/4DSTATS] y [/4DHMLSTATS] también permiten obtener información sobre el estado de la caché.

## Carpeta de certificados

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<code>certificateFolder</code>	La propiedad Text, pero puede ser un objeto <a href="#">4D.Folder</a> cuando se utiliza con el parámetro <i>settings</i> de la función <code>start()</code> .

Carpeta donde se encuentran los archivos del certificado TLS para el servidor web.

Por defecto con 4D o 4D Server, estos archivos deben colocarse junto a la [carpeta Project](#).

Con 4D en modo remoto, estos archivos deben estar ubicados en la carpeta de recursos locales de la base de datos en la máquina remota (ver [Carpeta base 4D Client](#) del comando [Get 4D folder](#)). Debe copiar estos archivos manualmente en la máquina remota.

Los archivos de certificados TLS son *key.pem* (documento que contiene la llave de cifrado privada) y *cert.pem* (documento que contiene el certificado).

## Conjunto de caracteres

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<code>characterSet</code>	Entero MIBEnum o cadena Name
<a href="#">WEB SET OPTION</a>	<code>Web character set</code>	Entero MIBEnum o cadena Name
Caja de diálogo de parámetros	<a href="#">Página Opciones (II) /Conjunto estándar</a>	Menú popup

Define el conjunto de caracteres que utilizará el servidor web de 4D. El valor por defecto depende del lenguaje del sistema operativo.

Esta configuración también se utiliza para generar informes rápidos en formato HTML.

## Lista de cifrado

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">cipherSuite</a>	Texto

Lista de cifrado utilizada para el protocolo seguro; establece la prioridad de los algoritmos de cifrado implementados por el servidor web. Puede ser una secuencia de cadenas separadas por dos puntos (por ejemplo "ECDHE-RSA-AES128-..."). Ver la [página de cifrados](#) en el sitio OpenSSL.

La lista de cifrado por defecto utilizada por 4D puede ser modificada para la sesión utilizando el comando [SET DATABASE PARAMETER](#), en cuyo caso la modificación se aplica a toda la aplicación 4D, incluyendo el servidor web, el servidor SQL, las conexiones cliente/servidor, así como el cliente HTTP y todos los comandos de 4D que hacen uso del protocolo seguro.

## Parámetros CORS

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">CORSSettings</a>	Colección de objetos (Lista de hosts y métodos permitidos para el servicio CORS)
<a href="#">WEB SET OPTION</a>	<a href="#">Web CORS settings</a>	Colección de objetos (Lista de hosts y métodos permitidos para el servicio CORS)
Caja de diálogo de parámetros	<a href="#">Options (II) page/Domain names and HTTP methods allowed</a>	Haga clic en el botón [+] para añadir un nombre de dominio permitido y su(s) método(s)

Lista de hosts y métodos permitidos para el servicio CORS.

#### Nombres de dominio (propiedad local)

Nombre de dominio o dirección IP desde donde las páginas externas pueden enviar solicitudes de datos al Servidor a través de CORS. Se pueden añadir múltiples atributos de dominio para crear una lista blanca. Se soportan varias sintaxis:

- 192.168.5.17:8081
- 192.168.5.17
- 192.168.\*
- 192.168.\*:8081
- <http://192.168.5.17:8081>
- [http://\\*.myDomain.com](http://*.myDomain.com)
- <http://myProject.myDomain.com>
- \*.myDomain.com
- myProject.myDomain.com
- \*

#### Métodos HTTP autorizados (propiedad métodos)

Métodos HTTP aceptados para el host CORS correspondiente. Se soportan los siguientes métodos HTTP:

- GET
- HEAD
- POST
- PUT
- DELETE
- OPTIONS
- TRACE
- PATCH

Separé cada método con un ";" (por ejemplo: "post;get"). Si methods está vacío, null o indefinido, todos los métodos están activos.

Ver también

[Activar CORS](#)

## Debug log

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">debugLog</a>	number
<a href="#">WEB SET OPTION</a>	<a href="#">Web debug log</a>	number

Status of the HTTP request log file of the web server ([HTTPDebugLog\\_nn.txt](#), stored in the "Logs" folder of the application -- nn is the file number). Es útil para depurar problemas relacionados con el servidor web. Registra cada

solicitud y cada respuesta en modo bruto. Se registran las solicitudes completas, incluidos los encabezados; opcionalmente, también se pueden registrar las partes del cuerpo.

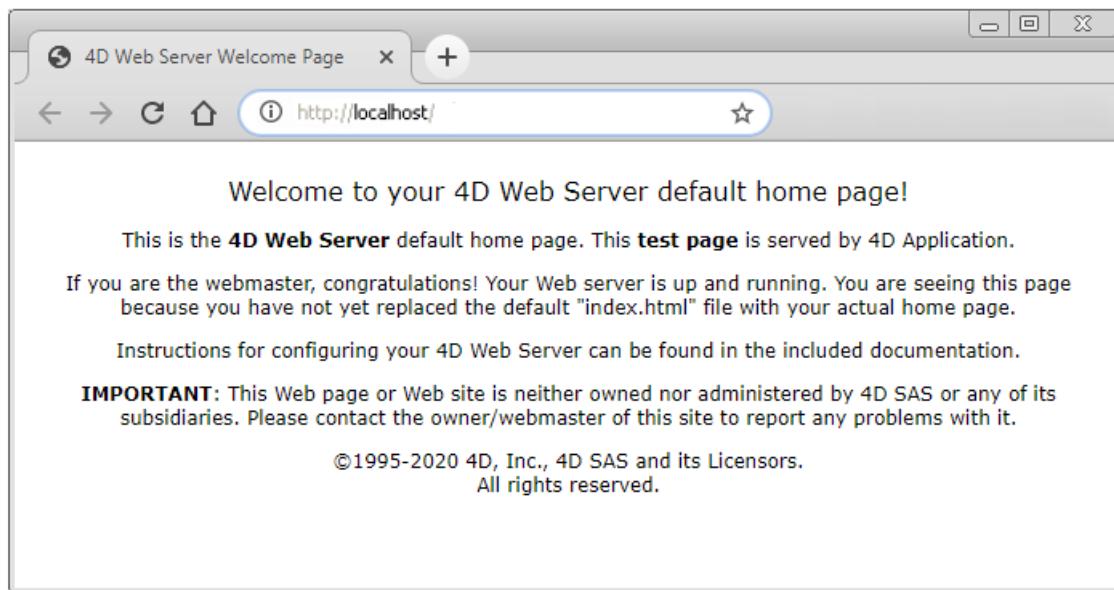
Valor	Constante	Descripción
0	wdl disable	Los debug logs Web HTTP son desactivados
1	wdl enable without body	Web HTTP debug log is enabled without body parts (body size is provided in this case)
3	wdl enable with response body	Web HTTP debug log is enabled with body part in response only
5	wdl enable with request body	Web HTTP debug log is enabled with body part in request only
7	wdl enable with all body parts	Web HTTP debug log is enabled with body parts in response and request

## Página de inicio por defecto

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">defaultHomepage</a>	Texto
WEB SET HOME PAGE		Puede ser diferente para cada proceso web
Caja de diálogo de parámetros	<a href="#">Página Configuración/Página de bienvenida por defecto</a>	

Designate a default home page for the web server. Esta página puede ser estática o [semi-dynamic].

By default, when the web server is launched for the first time, 4D creates a home page named "index.html" and puts it in the HTML root folder. If you do not modify this configuration, any browser connecting to the web server will obtain the following page:



You can designate another default home page by entering its pathname.

- La ruta es relativa a la [carpeta HTML raíz](#),
- La ruta se expresa con la sintaxis POSIX (las carpetas se separan con una barra ("/"))
- La ruta no debe comenzar ni terminar con una barra.

For example, if you want the default home page to be "MyHome.htm", and it is located in the "Web" folder (itself located in the default HTML root folder), use "Web/MyHome.htm".

If you do not specify any default home page, the `On Web Connection` database method is called. It is up to you to process the request procedurally.

## Activar CORS

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<code>CORSEnabled</code>	Booleano, true para activar CORS (False por defecto)
WEB SET OPTION	<code>Web CORS enabled</code>	0 (desactivado, por defecto) o 1 (activado)
Caja de diálogo de parámetros	<a href="#">Página Options (II)/Activar CORS</a>	Sin marcar por defecto

The 4D web server implements cross-origin resource sharing (CORS) to allow specific Web pages served from another domain to access the current Web application's resources via XHR calls, e.g., using REST. Por razones de seguridad, las peticiones "cross-domain" están prohibidas por defecto a nivel del navegador. When enabled, XHR calls (e.g. REST requests) from Web pages outside the domain can be allowed in your application (you need to define the list of allowed addresses in the CORS domain list, see CORS Settings below). In this case, if a non-allowed domain or method sends a cross site request, it is rejected with a "403 - forbidden" error response.

When disabled (default), all cross site requests sent with CORS are ignored.

For more information about CORS, please refer to the [Cross-origin resource sharing page](#) on Wikipedia.

Ver también

[Parámetros CORS](#)

## Activar HTTP

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<code>HTTPEnabled</code>	booleano
WEB SET OPTION	<code>Web HTTP enabled</code>	
Caja de diálogo de parámetros	<a href="#">Configuración/Activar HTTP</a>	

Indicates whether or not the web server will accept non-secure connections.

## Activar HTTPS

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<code>HTTPSEnabled</code>	booleano
WEB SET OPTION	<code>Web HTTPS enabled</code>	
Caja de diálogo de parámetros	<a href="#">Página configuración/Activar HTTPS</a>	

Estado de la comunicación a través de HTTPS. Esta opción se describe en [esta sección](#).

## Activar HSTS

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<code>HSTSEnabled</code>	Booleano, true para activar HSTS (por defecto es false)
WEB SET OPTION	<code>Web HSTS enabled</code>	0 (desactivado, por defecto) o 1 (activado)

## Estado de HTTP Strict Transport Security (HSTS).

When [HTTPS is enabled](#), keep in mind that if [HTTP is also enabled](#), the browser can still switch between HTTPS and HTTP (for example, in the browser URL area, the user can replace "https" by "http"). To forbid http redirections, you can [disable HTTP](#), however in this case an error message is displayed to client HTTP requests.

HSTS allows the 4D web server to declare that browsers should only interact with it via secure HTTPS connections. Once activated, the 4D web server will automatically add HSTS-related information to all response headers. Browsers will record the HSTS information the first time they receive a response from the 4D web server, then any future HTTP requests will automatically be transformed into HTTPS requests. The length of time this information is stored by the browser is specified with the Web HSTS max age setting.

HSTS requiere que HTTPS esté [activado](#) en el servidor. El [HTTP](#) también debe estar activado para permitir las conexiones iniciales del cliente.

Puede obtener el modo de conexión actual utilizando el comando [WEB Is secured connection](#).

## HSTS Max Age

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">HSTSMaxAge</a>	número en segundos
WEB SET OPTION	<a href="#">Web HSTS max age</a>	número en segundos

Specifies the maximum length of time (in seconds) that HSTS is active for each new client connection. Esta información se almacena del lado del cliente durante el tiempo especificado. El valor por defecto es 63072000 (2 años)

Atención: una vez activado HSTS, las conexiones de los clientes seguirán utilizando este mecanismo durante el tiempo especificado. Cuando esté probando sus aplicaciones, se recomienda definir una duración corta para poder cambiar entre los modos de conexión segura y no segura si es necesario.

## Nivel de compresión

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">HTTPCompressionLevel</a>	
WEB SET OPTION	<a href="#">Web HTTP compression level</a>	Se aplica a la Web y al servicio Web

Compression level for all compressed HTTP exchanges for the 4D web server (client requests or server replies). This setting lets you optimize exchanges by either privileging speed of execution (less compression) or the amount of compression (less speed). The choice of a value depends on the size and type of data exchanged.

Pass 1 to 9 as value where 1 is the fastest compression and 9 the highest. You can also pass -1 to get a compromise between speed and rate of compression. By default, the compression level is 1 (faster compression).

## Umbral de compresión HTTP

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">HTTPCompressionThreshold</a>	
WEB SET OPTION	<a href="#">Web HTTP compression threshold</a>	

In the framework of optimized HTTP exchanges, size threshold for requests below which exchanges should not be compressed. Este parámetro es útil para evitar la pérdida de tiempo de la máquina al comprimir los intercambios pequeños.

Pasa el tamaño expresado en bytes como valor. By default, the compression threshold is set to 1024 bytes.

## Puerto HTTP

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">HTTPPort</a>	number
WEB SET OPTION	Web port ID	
Caja de diálogo de parámetros	<a href="#">Página Configuración/Puerto HTTP</a>	

Número de puerto IP (TCP) de escucha para HTTP. By default, 4D publishes a web application on the regular Web HTTP Port (TCP port), which is port 80. If that port is already used by another web service, you need to change the HTTP Port used by 4D for this database.

En macOS, la modificación del puerto HTTP permite iniciar el servidor web 4D sin ser el usuario raíz de la máquina (ver [macOS HelperTool](#)).

From a web browser, you need to include the non-default HTTP port number into the address you enter for connecting to the web application. The address must have a suffix consisting of a colon followed by the port number. For example, if you are using the HTTP port number 8080, you will specify "123.4.567.89:8080".

Atención: si utiliza números de puerto TCP distintos a los predeterminados (80 para HTTP estándar y 443 para HTTPS), tenga cuidado de no utilizar números de puerto que sean predeterminados para otros servicios que pueda querer utilizar simultáneamente. Por ejemplo, si también tiene previsto utilizar el protocolo FTP en su equipo servidor web, no utilice los puertos TCP 20 y 21, que son los puertos por defecto para ese protocolo. Los números de puertos inferiores a 256 están reservados para servicios conocidos y los números de puertos de 256 a 1024 están reservados para servicios específicos originados en las plataformas UNIX. Para obtener la máxima seguridad, especifique un número de puerto más allá de estos intervalos (por ejemplo, en los 2000 o 3000).

If you specify 0, 4D will use the default HTTP port number 80.

## HTTP Trace

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">HTTPTrace</a>	Booleano, falso por defecto
WEB SET OPTION	Web HTTP TRACE	Integer, 0 por defecto (desactivado)

Activación del método HTTP TRACE en el servidor web 4D. For security reasons, by default the 4D web server rejects HTTP TRACE requests with an error 405. If necessary, you can enable the HTTP TRACE method, in which case the 4D Web server replies to HTTP TRACE requests with the request line, header, and body.

## Puerto HTTPS

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">HTTPSSPort</a>	number

| WEB SET OPTION | Web HTTPS port ID ||

|Settings dialog box|[Configuration page/HTTPS Port](#)||

Listening IP port number for HTTPS connections via TLS. Por defecto, el valor es 443 (valor estándar). See also [HTTP Port](#) for information on port numbers.

## Tiempo de espera del proceso inactivo

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">inactiveProcessTimeout</a>	
WEB SET OPTION	Web inactive process timeout	
Caja de diálogo de parámetros	<a href="#">Página Opciones (I)/Tiempo de espera de procesos inactivos</a>	Cursor

Life duration (in minutes) of inactive processes associated with sessions. At the end of the timeout, the process is killed on the server, the `On Web Close Process` database method is called, then the session context is destroyed.

Default: 480 minutes (pass 0 to restore the default value)

## Tiempo de espera de las sesiones inactivas

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">inactiveSessionTimeout</a>	
WEB SET OPTION	Web inactive session timeout	

Life duration (in minutes) of inactive sessions (duration set in cookie). Al final de este periodo, la cookie de sesión expira y deja de ser enviada por el cliente HTTP.

Default: 480 minutes (pass 0 to restore the default value)

## Dirección IP de escucha

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">IPAddressToListen</a>	
WEB SET OPTION	Web IP address to listen	
Caja de diálogo de parámetros	<a href="#">Página Configuración/Dirección IP</a>	Menú popup

IP address strings on which the 4D web server will receive HTTP requests (4D local and 4D Server).

By default, no specific address is defined ( Any value in the Settings dialog box), which means that the server responds to all IP addresses. When you designate a specific address, the server only responds to requests sent to this address. This feature is designed for 4D web servers located on machines with multiple TCP/IP addresses. It is, for example, frequently the case of most host providers.

Valores posibles: cadena de direcciones IP. Both IPv6 string formats (e.g.

"2001:0db8:0000:0000:ff00:0042:8329") and IPv4 string formats (e.g. "123.45.67.89") are supported.

### Acerca de la compatibilidad con IPv6

- No hay aviso cuando el puerto TCP está ocupado

Cuando el servidor está configurado para responder en las direcciones IP "Todas", si el puerto TCP está siendo utilizado por otra aplicación, esto no se indica cuando se inicia el servidor. De hecho, el servidor 4D no detecta ningún error en este caso porque el puerto permanece libre en la dirección IPv6. Sin embargo, no es posible acceder a ella utilizando la dirección IPv4 de la máquina, ni mediante la dirección local 127.0.0.1.

Si su servidor 4D no parece responder en el puerto definido, puede probar la dirección `[::1]` en la máquina del servidor (equivalente a 127.0.0.1 para IPv6, añada `[:portNum]` para probar otro número de puerto). Si 4D responde, es probable que otra aplicación esté utilizando el puerto en IPv4.

- Direcciones IPv6 mapeadas en IPv4

Para estandarizar el procesamiento, 4D ofrece una representación híbrida estándar de las direcciones IPv4 en IPv6. Estas direcciones se escriben con un prefijo de 96 bits en formato IPv6, seguido de 32 bits escritos en la notación decimal punto de IPv4. Por ejemplo, `::ffff:192.168.2.34` representa la dirección IPv4 192.168.2.34.

- Indicación de los números de puerto

Dado que la notación IPv6 utiliza dos puntos (:), la adición de números de puerto puede dar lugar a cierta confusión, por ejemplo:

```
2001:0DB8::85a3:0:ac1f:8001 // dirección IPv6
2001:0DB8::85a3:0:ac1f:8001:8081 // dirección IPv6 con puerto 8081
```

To avoid this confusion, we recommend using the [ ] notation whenever you combine an IPv6 address with a port number, for instance:

```
[2001:0DB8::85a3:0:ac1f:8001]:8081 //Dirección IPv6 con puerto 8081
```

## Sesiones Keep

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">keepSession</a>	
WEB SET OPTION	<a href="#">Web keep session</a>	
Caja de diálogo de parámetros	<a href="#">Options (I) page/Legacy sessions (single process sessions)</a>	sólo en los proyectos convertidos

Legacy session management enabling status for the 4D web server (deprecated).

Cuando esta opción está marcada, la opción "Reutilización de los contextos temporales" se marca automáticamente (y se bloquea).

## Registro de los logs

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">logRecording</a>	
WEB SET OPTION	<a href="#">Web log recording</a>	
Caja de diálogo de parámetros	<a href="#">Página log (tipo)</a>	Menú popup

Starts or stops the recording of requests received by the 4D web server in the *logweb.txt* file and sets its format. By default, requests are not recorded (0/No Log File). When enabled, the *logweb.txt* file is automatically placed in the Logs folder.

This setting allows you to select the format of this file. Valores disponibles:

Valor	Nombre del formato	Descripción
0	No hay archivo de historial	Por defecto
1	Registro en formato CLF	Formato de historial común - Cada línea del archivo representa una petición, como: host rfc931 user [DD/MMM/YYYY:HH:MM:SS] "request" state length - Cada campo está separado por un espacio y cada línea termina con la secuencia CR/LF.
2	Registro en formato DLF	Combined Log Format - Similar al formato CLF, pero añade dos campos HTTP adicionales al final de cada solicitud: Referer y User-agent.
3	Registro en formato ELF	Extended Log Format - A personalizar en la caja de diálogo de las Propiedades
4	Registro en formato WLF	WebStar Log Format - A personalizar en la caja de diálogo de las Propiedades

Formats 3 and 4 are custom formats whose contents must be set beforehand in the [Settings dialog box](#). Si utiliza uno de estos formatos sin haber seleccionado ninguno de sus campos en esta página, el archivo de registro no se generará.

## Procesos Web simultáneos máximos

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">maxConcurrentProcesses</a>	
WEB SET OPTION	Web max concurrent processes	
Caja de diálogo de parámetros	<a href="#">Options (I) page/Maximum Concurrent Web Processes</a>	

Strictly high limit of concurrent web processes that can be simultaneously open on the server when no sessions or legacy sessions are used (scalable sessions support an [unlimited number](#) of preemptive processes). This parameter allows prevention of server saturation as the result of massive number of requests. When the maximum number of concurrent Web processes (minus one) is reached, 4D no longer creates new processes and sends the HTTP status [503 – Service Unavailable](#) to all new requests.

Por defecto, el valor es 100. You can set the number anywhere between 10 and 32000.

## Tamaño máximo de la petición

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">maxRequestSize</a>	
WEB SET OPTION	Web maximum requests size	

Maximum size (in bytes) of incoming HTTP requests (POST) that the web server is authorized to process. By default, the value is 2 000 000, i.e. a little less than 2 MB. Passing the maximum value (2 147 483 648) means that, in practice,

no limit is set.

Este límite se utiliza para evitar la saturación del servidor web debido a peticiones entrantes demasiado grandes. Este límite se utiliza para evitar la saturación del servidor web debido a peticiones entrantes demasiado grandes.

Valores posibles: 500 000 a 2 147 483 648.

## Número máximo de sesiones

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">maxSessions</a>	
WEB SET OPTION	Web max sessions	

Número máximo de sesiones simultáneas. When you reach the limit set, the oldest session is closed (and [On Web Close Process](#) database method is called) if the Web server needs to create a new one. The number of simultaneous sessions cannot exceed the [maximum number of Web processes](#) (100 by default).

Default value: 100 (pass 0 to restore the default value).

## Versión TLS mínima

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">minTLSVersion</a>	number

Versión mínima de TLS aceptada para las conexiones. Se rechazarán los intentos de conexión de clientes que sólo soporten versiones inferiores a la mínima.

Valores posibles:

- 1 = TLSv1\_0
- 2 = TLSv1\_1
- 3 = TLSv1\_2 (por defecto)
- 4 = TLSv1\_3

Valores posibles:

La versión TLS mínima utilizada por 4D puede ser modificada para la sesión utilizando el comando [SET DATABASE PARAMETER](#), en cuyo caso la modificación se aplica a toda la aplicación 4D, incluyendo el servidor web, el servidor SQL y las conexiones cliente/servidor.

## Nombre

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">name</a>	

Nombre de la aplicación del servidor web. Útil cuando se inician los servidores web de los componentes.

## Versión OpenSSL

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">openSSLVersion</a>	Sólo lectura

Versión de la librería OpenSSL utilizada.

## Perfect Forward Secrecy

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">perfectForwardSecrecy</a>	Booleano, de sólo lectura

True if PFS is available on the web server (see [TLS](#) section).

## Reutilizar los contextos temporales (en modo remoto)

Puede ajustarse con	Nombre	Comentarios
Caja de diálogo de parámetros	<a href="#">Options (I) page/Maximum Concurrent Web Processes</a>	

This option is only available when No sessions option is checked.

Allows you to optimize the operation of the 4D Web Server in remote mode by reusing web processes created for processing previous web requests. In fact, the web server in 4D needs a specific web process for the handling of each web request; in remote mode, when necessary, this process connects to the 4D Server machine in order to access the data and database engine. It thus generates a temporary context using its own variables, selections, etc. Once the request has been dealt with, this process is killed.

When the Reuse Temporary Contexts option is checked, in remote mode 4D maintains the specific web processes and reuses them for subsequent requests. By removing the process creation stage, web server performance is improved.

In return, you must make sure in this case to systematically initialize the variables used in 4D methods in order to avoid getting incorrect results. Similarly, it is necessary to erase any current selections or records defined during the previous request.

This option only has an effect with a 4D web server in remote mode. With a 4D in local mode, all web processes (other than session processes) are killed after their use.

## Robots.txt

Certain robots (query engines, spiders...) scroll through web servers and static pages. If you do not want robots to be able to access your entire site, you can define which URLs they are not allowed to access.

To do so, put the ROBOTS. TXT file at the server's root. This file must be structured in the following manner:

```
User-Agent: <name>
Disallow: <URL> o <beginning of the URL>
```

Por ejemplo:

```
User-Agent: *
Disallow: /4D
Disallow: /%23%23
Disallow: /GIFS/
```

- “User-Agent: \*” - todos los robots son afectados.
- “Disallow: /4D” - Los robots no están autorizados a acceder a los URLs comenzando por /4D.
- “Disallow: /%23%23” - Los robots no están autorizados a acceder a los URLs comenzando por /%23%23.
- “Disallow: /GIFS/” - Los robots no pueden acceder a la carpeta /GIFS/ ni a sus subcarpetas.

Otro ejemplo:

```
User-Agent: *
Disallow: /
```

In this case, robots are not allowed to access the entire site.

## Root Folder

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">rootFolder</a>	Text property but can be a <a href="#">4D.Folder</a> object when used with the <i>settings</i> parameter of the <code>start()</code> function
<code>WEB SET ROOT FOLDER</code>		
Caja de diálogo de parámetros	<a href="#">Página Configuración/Raíz HTML por defecto</a>	

Path of web server root folder, i.e. the folder in which 4D will search for the static and semi-dynamic HTML pages, pictures, etc., to send to the browsers. La ruta de acceso está en formato POSIX (ruta completa). The web server will need to be restarted in order for the new root folder to be taken into account.

Moreover, the HTML root folder defines, on the web server hard drive, the hierarchical level above which the files will not be accessible. If a requested URL or a 4D command tries to access a file located above the HTML root folder, an error is returned indicating that the file has not been found.

By default, 4D defines a HTML Root folder named `WebFolder`. If it does not already exist, the HTML root folder is physically created on disk at the moment the Web server is launched for the first time. Se crea la carpeta raíz:

- con 4D (local) y 4D Server, en el mismo nivel de la [carpeta del proyecto](#).
- con 4D en modo remoto, en la carpeta de recursos locales.

You can designate another default HTML root folder by entering its pathname.

- La ruta es relativa a la [carpeta del proyecto](#) (4D local y 4D Server) o a la carpeta que contiene la aplicación 4D o el paquete de software (4D en modo remoto).
- La ruta se expresa con la sintaxis POSIX (las carpetas se separan con una barra ("/"))
- Para "subir" un nivel en la jerarquía de las carpetas, introduzca `..` (dos puntos) antes del nombre de la carpeta
- La ruta no debe comenzar con una barra (excepto si quiere que la carpeta raíz HTML sea la carpeta remota del proyecto o de 4D, pero que el acceso a las carpetas anteriores esté prohibido, en cuyo caso puede pasar "/" como carpeta raíz).

For example, if you want the HTML root folder to be the "Web" subfolder in the "MyWebApp" folder, enter "MyWebApp/Web".

When the HTML root folder is modified, the cache is cleared so as to not store files whose access is restricted.

## Sesiones escalables

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<a href="#">scalableSession</a>	
<code>WEB SET OPTION</code>	<a href="#">Sesión escalable web</a>	
Caja de diálogo de parámetros	<a href="#">Options (I) page/Scalable sessions (multi-process sessions)</a>	

Scalable session management enabling status for the 4D web server. Web server sessions are detailed in the [User](#)

sessions page.

## Session Cookie Domain

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<code>sessionCookieDomain</code>	
WEB SET OPTION	Web session cookie domain	

Campo "path" de la cookie de sesión. Useful for controlling the scope of the session cookies. Si define, por ejemplo, el valor "/\*.4d.fr" para este selector, el cliente sólo enviará una cookie cuando la solicitud se dirija al dominio ".4d.fr", lo que excluye a los servidores que alojan datos estáticos externos.

## Nombre de la cookie de sesión

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<code>sessionCookieName</code>	
WEB SET OPTION	Web session cookie name	

Nombre de la cookie utilizada para guardar el ID de sesión. Por defecto = "4DSID".

## Ruta de la cookie de sesión

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<code>sessionCookiePath</code>	
WEB SET OPTION	Web session cookie path	

Campo "path" de la cookie de sesión. Se utiliza para controlar el alcance de las cookies de sesión. Si define, por ejemplo, el valor "/4DACTION" para este selector, el cliente sólo enviará una cookie para las peticiones dinámicas que empiecen por 4DACTION, y no para las imágenes, páginas estáticas, etc.

## Session Cookie SameSite

Puede ajustarse con	Nombre	Comentarios
objeto webServer	<code>sessionCookieSameSite</code>	

Value of the `SameSite` attribute value of the session cookie. This attribute allows you to declare if your cookie should be restricted to a first-party or same-site context, as a protection from some cross-site request forgery ([CSRF](#)) attacks.

For a detailed description of the `SameSite` attribute, please refer to the [Mozilla documentation](#) or [this web.dev page](#).

Hay tres valores disponibles:

- "Estricto" (valor predeterminado del atributo `SameSite` para las cookies de sesión de 4D): las cookies sólo se enviarán en el contexto de primera parte, es decir, el contexto correspondiente al dominio del sitio y nunca a sitios web de terceros.
- "Lax": las cookies no se envían en las subpeticiones de sitios cruzados (por ejemplo, para cargar imágenes o marcos en un sitio de terceros), sino que se envían cuando un usuario está navegando hacia el sitio de origen (es decir, sigue un enlace).
- "Ninguna": las cookies se envían en todos los contextos, es decir, en las respuestas a las solicitudes de primera parte y de origen cruzado. Cuando se utiliza el valor "None", el atributo cookie `Secure` también debe ser definido (o

la cookie será bloqueada).

The `Secure` attribute value of the session cookie is automatically set to "True" if the connection is HTTPS (whatever the `SameSite` attribute value).

It is not recommended to set `SameSite=None` on a HTTP server since the `Secure` attribute will be missing (used in HTTPS only) and cookies will be blocked.

## Utilizar los procesos apropiativos

Puede ajustarse con	Nombre	Comentarios
Caja de diálogo de parámetros	<a href="#">Options (I) page/Maximum Concurrent Web Processes</a>	

This option enables the preemptive mode for your application's web server code when `No sessions` option is selected (the preemptive mode is always enabled with scalable sessions). When this option is checked in this context, the 4D compiler will automatically evaluate the thread-safety property of each piece of [web-related code](#) and return errors in case of incompatibility.

## Parámetros obsoletos

The following settings are still supported but rely on deprecated features or technologies. Generalmente se recomienda mantener los valores por defecto.

### Autorizar el acceso a la base de datos a través de las URL 4DSYNC

This option controls the support of HTTP synchronization requests containing deprecated `/4DSYNC` URLs.

### Session IP Address Validation

This option is not available in [scalable sessions mode](#) (there is no validation).

Estado de validación de la dirección IP para las cookies de sesión. For security reasons, by default the 4D web server checks the IP address of each request containing a session cookie and rejects it if this address does not match the IP address used to create the cookie. En algunas aplicaciones específicas, es posible que desee desactivar esta validación y aceptar las cookies de sesión, incluso cuando sus direcciones IP no coinciden. For example when mobile devices switch between Wifi and 4G/5G networks, their IP address will change. In this case, you must pass 0 in this option to allow clients to be able to continue using their Web sessions even when the IP addresses change. Note that this setting lowers the security level of your application. When it is modified, this setting is effective immediately (you do not need to restart the HTTP server).

### Enviar directamente los caracteres extendidos

When this option is checked, the web server sends extended characters "as is" in semi-dynamic pages, without converting them into HTML entities. This option has shown a speed increase on most foreign operating systems (especially the Japanese system).

### Conexiones Keep-Alive

El servidor web 4D puede utilizar conexiones persistentes. The keep-alive option allows you to maintain a single open TCP connection for the set of exchanges between the web browser and the server to save system resources and to optimize transfers.

The Use Keep-Alive Connections option enables or disables keep-alive TCP connections for the web server. Esta opción está activada por defecto. In most cases, it is advisable to keep this option check since it accelerates the exchanges. If the web browser does not support connection keep alive, the 4D Web Server automatically switches to HTTP/1.0.

The 4D Web Server keep-alive function concerns all TCP/IP connections (HTTP, HTTPS). Note however that keep-alive connections are not always used for all 4D web processes.

In some cases, other optimized internal functions may be invoked. Keep-alive connections are useful mainly for static pages.

Two options allow you to set how the keep-alive connections work:

- Número de peticiones por conexión: permite definir el número máximo de peticiones y de respuestas capaces de viajar por una conexión persistente. Limitar el número de peticiones por conexión permite evitar la inundación del servidor debido a un gran número de peticiones entrantes (una técnica utilizada por los hackers).

El valor por defecto (100) puede ser aumentado o disminuido en función de los recursos de la máquina que aloja el servidor 4D web.

- Tiempo de espera antes de desconexión : este valor define el periodo máximo de espera (en segundos) durante el cual el servidor web mantiene una conexión TCP abierta sin recibir ninguna petición del navegador web. Una vez finalizado este periodo, el servidor cierra la conexión.

Si el navegador web envía una solicitud después de cerrar la conexión, se crea automáticamente una nueva conexión TCP. Esta operación no es visible para el usuario.

# Administración

4D ofrece varias herramientas integradas para iniciar, detener o supervisar el servidor web integrado.

## Iniciar el servidor Web 4D

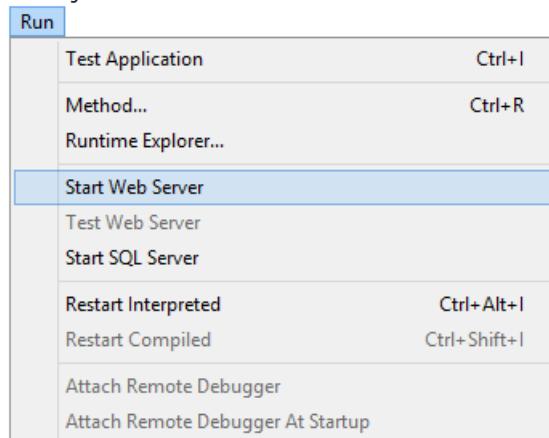
Para poder lanzar el servidor web de 4D o 4D Server, debe tener una licencia "4D Web Application". Para más información, consulte el [sitio web de 4D](#).

Un proyecto 4D puede iniciar y monitorizar un servidor web para la aplicación principal (host) así como para cada componente alojado.

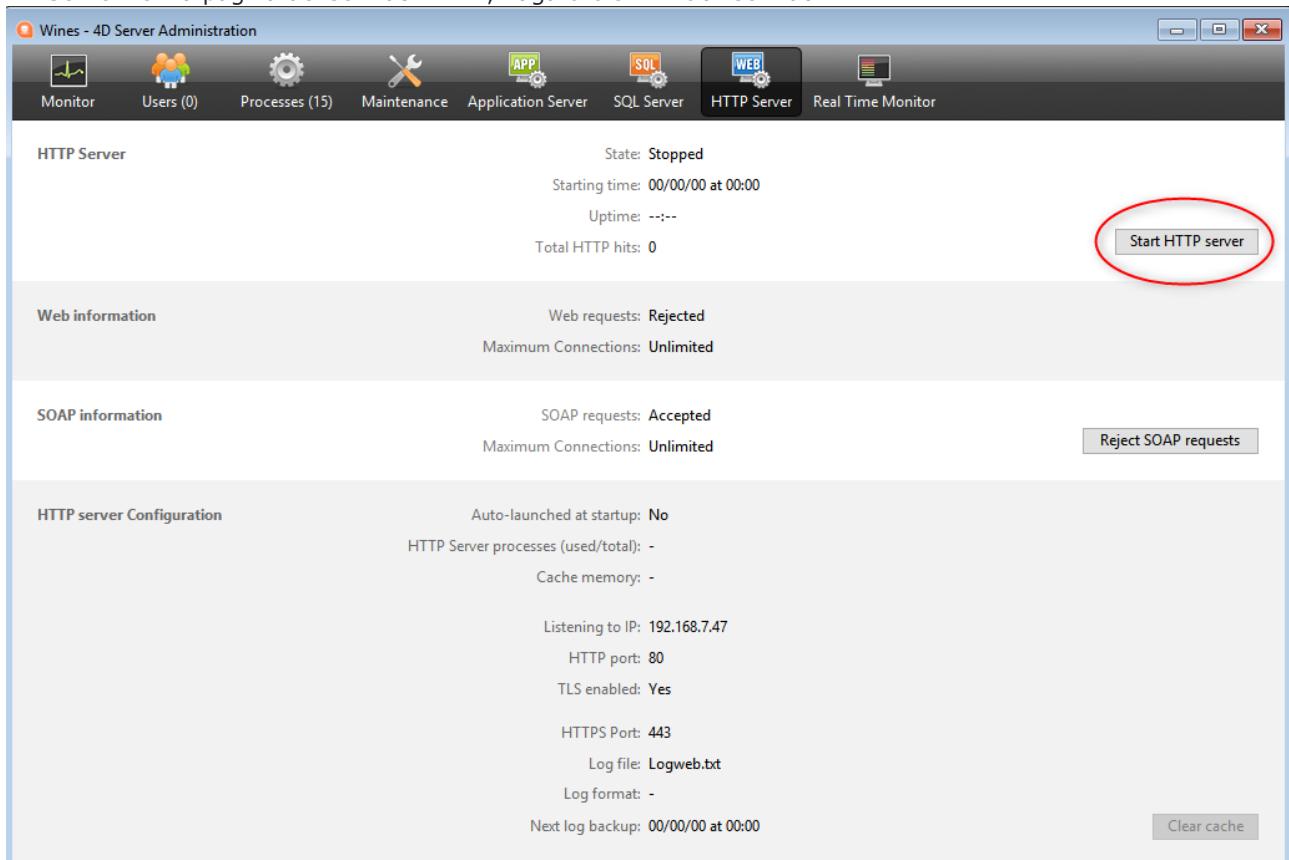
El servidor web principal de 4D puede iniciarse de diferentes maneras:

- Utilizando un botón o comando de menú.

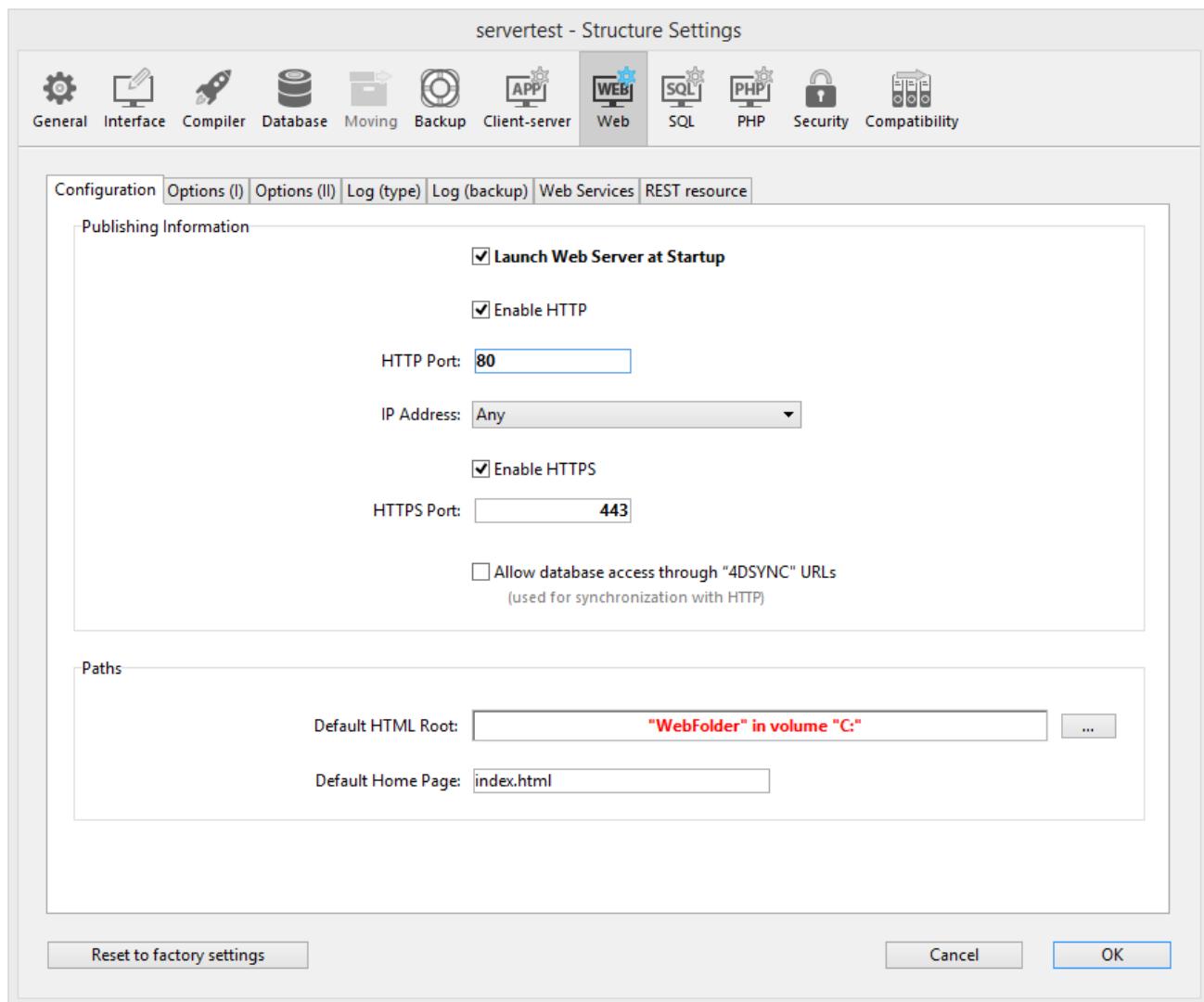
- 4D: Ejecución > iniciar el servidor web



- 4D Server: en la página del servidor HTTP, haga clic en Iniciar servidor HTTP



- Se inicia automáticamente cada vez que se abre la aplicación 4D. Para ello, despliegue la página Web/Configuración de la Configuración y seleccione la casilla Lanzar el servidor web al inicio:



- Por programación, llamando la función `webServer.start()` o el comando `WEB START SERVER`.

El servidor web de cualquier componente puede iniciarse llamando a la función `webServer.start()` en el objeto servidor web del componente.

No es necesario relanzar la aplicación 4D para iniciar o detener el servidor web.

## Detener el servidor Web 4D

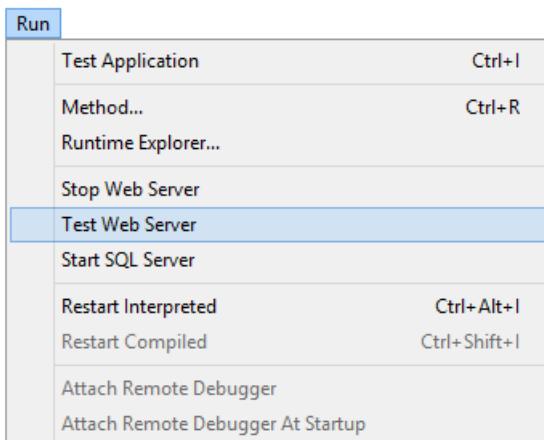
El servidor web principal de 4D puede detenerse de diferentes maneras:

- Utilizando el menú 4D Ejecución>Detener el servidor Web, o vía el botón Detener el servidor HTTP de la página Servidor HTTP de 4D Server (ambos elementos muestran Iniciar... cuando el servidor no está ya iniciado).
- Por programación, llamando la función `webServer.stop()` o el comando `WEB STOP SERVER`.

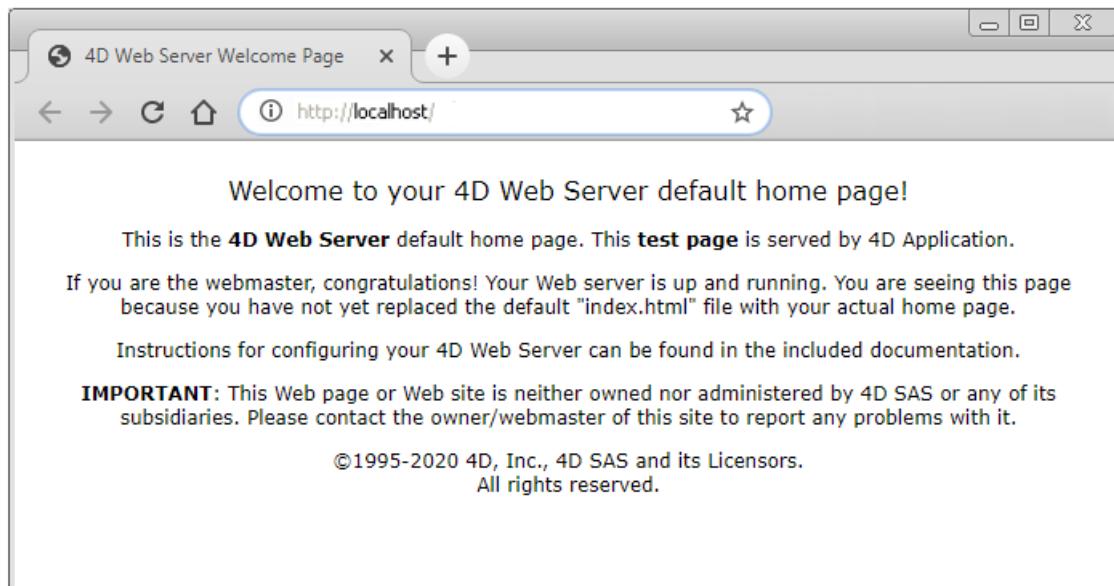
El servidor web de cualquier componente puede detenerse llamando a la función `webServer.stop()` en el objeto servidor web del componente.

## Probar el servidor Web 4D

El comando Test Web Server puede utilizarse para asegurarse de que el servidor web integrado funciona correctamente (sólo en 4D). Este comando es accesible en el menú Ejecutar cuando se lanza el servidor web:



Al seleccionar este comando, la página de inicio del sitio web publicado por la aplicación 4D se muestra en una ventana de su navegador web predeterminado:



Este comando permite verificar que el servidor web, la visualización de la página de inicio, etc. funcionan correctamente. La página se llama utilizando la URL *localhost*, que es el atajo estándar que designa la dirección IP de la máquina en la que se ejecuta el navegador web. El comando tiene en cuenta el número de [puerto de publicación TCP](#) especificado en los parámetros.

## Borrar la caché

En cualquier momento, puede vaciar la caché de las páginas y de las imágenes que contiene (si, por ejemplo, ha modificado una página estática y quiere volver a cargarla en la caché).

Para ello, puede:

- 4D: click on the Clear Cache button in the Web/Options (I) page of the Settings dialog box.
- 4D Server: click on the Clear Cache button in the HTTP page of the [4D Server Administration window](#).

La caché se borra inmediatamente.

También puede utilizar la URL </4DCACHECLEAR>.

## Explorador de ejecución

The Watch page (Web heading) in the Runtime Explorer displays web server information, particularly:

- Web Cache Usage: indicates the number of pages present in the web cache as well as its use percentage. This information is only available if the web server is active and if the cache size is greater than 0.

- Web Server Elapsed Time : indicates the duration of use (in hours:minutes:seconds format) of the Web server. This information is only available if the web server is active.
- Web Hits Count : indicates the total number of HTTP requests received since the web server boot, as well as an instantaneous number of requests per second (measure taken between two Runtime Explorer updates). This information is only available if the web server is active.

## URL para la administración

Website administration URLs allow you to control the website published on your server. 4D Web Server accepts four particular URLs: `/4DSTATS`, `/4DHTMLSTATS`, `/4DCACHECLEAR` and `/4WEBTEST`.

`/4DSTATS`, `/4DHTMLSTATS` y `/4DCACHECLEAR` sólo están disponibles para el diseñador y el administrador de la base de datos. Si el sistema de contraseñas 4D no ha sido activado, estas URLs están disponibles para todos los usuarios. `/4WEBTEST` está siempre disponible.

### /4DSTATS

The `/4DSTATS` URL returns several items of information in an HTML table (displayable in a browser):

Elemento	Descripción
Tamaño actual de la caché	Tamaño actual de la caché del servidor web (en bytes)
Tamaño máximo de la caché	Tamaño máximo de la caché (en bytes)
Cached Object Max Size	Tamaño máximo de cada objeto en la caché (en bytes)
Cache Use	Porcentaje de caché utilizado
Cached Objects	Número de objetos encontrados en la caché, incluyendo imágenes

This information can allow you to check the functioning of your server and eventually adapt the corresponding parameters.

El comando `WEB GET STATISTICS` permite obtener también información sobre cómo se está utilizando la caché de las páginas estáticas.

### /4DHTMLSTATS

The `/4DHTMLSTATS` URL returns, also as an HTML table, the same information as the `/4DSTATS` URL. The difference is that the Cached Objects field only counts HTML pages (without counting picture files). Moreover, this URL returns the Filtered Objects field.

Elemento	Descripción
Tamaño actual de la caché	Tamaño actual de la caché del servidor web (en bytes)
Tamaño máximo de la caché	Tamaño máximo de la caché (en bytes)
Cached Object Max Size	Tamaño máximo de cada objeto en la caché (en bytes)
Cache Use	Porcentaje de caché utilizado
Cached Objects	Número de objetos encontrados en la caché, sin imágenes
Objetos filtrados	Número de objetos en la caché que no se cuentan por URL, en particular, las imágenes

### /4DCACHECLEAR

The `/4DCACHECLEAR` URL immediately clears the cache of the static pages and images. It allows you to therefore "force" the update of the pages that have been modified.

## /4DWEBTEST

The `/4DWEBTEST` URL is designed to check the web server status. When this URL is called, 4D returns a text file with the following HTTP fields filled:

Campo HTTP	Descripción	Ejemplo
Fecha	fecha actual en el formato RFC 822	Mon, 7 Dec 2020 13:12:50 GMT
Server	4D/Número de versión	4D/18.5.0 (Build 18R5.257368)
User-Agent	nombre y versión @ dirección IP del cliente	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.183 Safari/537.36 @ 127.0.0.1

## Logs

4D allows you to generate two logs of web requests:

- a debug log, useful in the web server development phase (`HTTPDebugLog.txt`),
- a standardized web request log, rather used for statistic purposes (`logweb.txt`).

Both log files are automatically created in the Logs folder of the application project.

### HTTPDebugLog.txt

The [http debug file](#) can be enabled using the `web server object` or the `WEB SET OPTION` command.

Este archivo de historial registra cada petición HTTP y cada respuesta en modo crudo. Se registran las solicitudes completas, incluidos los encabezados; opcionalmente, también se pueden registrar las partes del cuerpo.

Los siguientes campos se registran tanto para la solicitud como para la respuesta:

Nombre del campo	Descripción
SocketID	ID del socket utilizado para la comunicación
PeerIP	Dirección IPv4 del host (cliente)
PeerPort	Puerto utilizado por el host (cliente)
TimeStamp	Timestamp en milisegundos (desde el inicio del sistema)
ConnectionID	Conexión UUID (UUID del VTCPSocket utilizado para la comunicación)
SequenceNumber	Número de operación único y secuencial en la sesión de historial

### logweb.txt

The [web log recording file](#) can be enabled using the `web server object`, the `WEB SET OPTION` command, or the Web/Log (type) page of the settings. Debe seleccionar el formato de historial.

### CLF/DLF

Each line of the file represents a request, such as: `host rfc931 user [DD/MMM/YYYY:HH:MM:SS] "request" state length`. Each field is separated by a space and each line ends by the CR/LF sequence (character 13, character 10).

DLF (Combined Log Format) format is similar to CLF (Common Log Format) format and uses exactly the same structure. It simply adds two additional HTTP fields at the end of each request: Referer and User-agent. Here is the description of CLF/DLF formats (not customizable):

Nombre del campo	Descripción
host	Dirección IP del cliente (por ejemplo: "192.100.100.10)
rfc931	información no generada por 4D, siempre es - (un signo menos)
user	nombre de usuario como está autenticado, o - (un signo menos). Si el nombre de usuario contiene espacios, se remplazan por _ (un guión bajo).
DD/MMM/YYYY:HH:MM:SS	DD: día, MMM: una abreviatura de 3 letras para el nombre del mes (Jan, Feb,...), YYYY: año, HH: hora, MM: minutos, SS: segundos. La fecha y hora son locales al servidor.
request	solicitud enviada por el cliente (por ejemplo, GET /index.htm HTTP/1.0)
state	respuesta dada por el servidor
length	tamaño de los datos devueltos (excepto el encabezado HTTP) o 0
Referer	Sólo DLF- Contiene la URL de la página que apunta al documento solicitado.
User-agent	DLF únicamente - Contiene el nombre y la versión del navegador o del software del cliente en el origen de la solicitud

## ELF/WLF

The ELF (Extended Log Format) format is very widespread in the world of HTTP browsers. It can be used to build sophisticated logs that meet specific needs. For this reason, the ELF format can be customized: it is possible to choose the fields to be recorded as well as their order of insertion into the file.

The WLF (WebStar Log Format) was developed specifically for the 4D WebSTAR server.

### Configurar los campos

When you choose the ELF or WLF format, the "Web Log Token Selection" area displays the fields available for the chosen format. You will need to select each field to be included in the log. Para ello, marque los campos deseados.

No puede seleccionar el mismo campo dos veces.

The following table lists the fields available for each format (in alphabetical order) and describes its contents:

Campo	ELF	WLF	Valor
BYTES_RECEIVED		X	Número de bytes recibidos por el servidor
BYTES_SENT	X	X	Number of bytes sent by the server to the client
C_DNS	X	X	IP address of the DNS (ELF: field identical to the C_IP field)
C_IP	X	X	IP address of the client (for example 192.100.100.10)
CONNECTION_ID		X	Número de identificación de la conexión
CS(COOKIE)	X	X	Information about cookies contained in the HTTP request
CS(HOST)	X	X	Campo Host de la petición HTTP
CS(REFERER)	X	X	URL de la página que apunta al documento solicitado
CS(USER_AGENT)	X	X	Information about the software and operating system of the client
CS_SIP	X	X	Dirección IP del servidor
CS_URI	X	X	URI sobre el que se realiza la petición
CS_URI_QUERY	X	X	Parámetros de consulta de la petición
CS_URI_STEM	X	X	Parte de la petición sin los parámetros de consulta
DATE	X	X	DD: day, MMM: 3-letter abbreviation for month (Jan, Feb, etc.), YYYY: year
METHOD	X	X	HTTP method used for the request sent to the server
PATH_ARGS		X	Parámetros CGI: cadena situada después del carácter "\$"
STATUS	X	X	Respuesta ofrecida por el servidor
TIME	X	X	HH: hour, MM: minutes, SS: seconds
TRANSFER_TIME	X	X	Tiempo solicitado por el servidor para generar la respuesta
USER	X	X	User name if authenticated; otherwise - (minus sign). If the user name contains spaces, they are replaced by _ (underlines)
URL		X	URL solicitado por el cliente

Las fechas y horas se indican en GMT.

## Frecuencia del backup

Dado que un archivo *logweb.txt* puede llegar a ser considerablemente grande, es posible establecer un mecanismo de archivo automático. La activación de una copia de seguridad puede basarse en un periodo de tiempo determinado (expresado en horas, días, semanas o meses), o en función del tamaño del archivo; cuando se alcanza el plazo establecido (o el tamaño del archivo), 4D cierra y archiva automáticamente el archivo de registro actual y crea uno nuevo.

Cuando se activa la copia de seguridad del archivo de registro web, el archivo de registro se archiva en una carpeta llamada "Archivos Logweb", que se crea en el mismo nivel que el archivo *logweb.txt*.

El fichero archivado se renombra según el siguiente ejemplo "DYY\_MM\_DD\_Thh\_mm\_ss.txt". Por ejemplo, para un fichero archivado el 4 de septiembre de 2020 a las 15:50. y 7 segundos: "D2020\_09\_04\_T15\_50\_07.txt."

## Parámetros de backup

Los parámetros de copia de seguridad automática del *logweb.txt* se definen en la página Web/Log (copia de seguridad) de los parámetros:

webServer - Structure Settings

Backup Frequency for Web Log File

- No Backup
- Every  hour(s) starting at
- Every  day(s) at
- Every  week(s)
  - Monday at
  - Tuesday at
  - Wednesday at
  - Thursday at
  - Friday at
  - Saturday at
  - Sunday at
- Every  month(s) Day  at
- Every

[Reset to factory settings](#) [Cancel](#) [OK](#)

Primero debe elegir la frecuencia (días, semanas, etc.) o el criterio de límite de tamaño de los archivos haciendo clic en el botón de opción correspondiente. A continuación, debe especificar el momento preciso de la copia de seguridad si es necesario.

- No Backup: The scheduled backup function is deactivated.
- Every X hour(s): This option is used to program backups on an hourly basis. Puede introducir un valor entre 1 y 24
  - starting at: Used to set the time at which the first back up will begin.
- Every X day(s) at X: This option is used to program backups on a daily basis. Introduzca 1 si desea realizar una copia de seguridad diaria. When this option is checked, you must indicate the time when the backup must be started.
- Every X week(s), day at X: This option is used to program backups on a weekly basis. Introduzca 1 si desea realizar una copia de seguridad semanal. Introduzca 1 si desea realizar una copia de seguridad semanal. When this option is checked, you must indicate the day(s) of the week and the time when each backup must be started. You can select several days of the week if desired.
- Every X month(s), Xth day at X: This option is used to program backups on a monthly basis. Introduzca 1 si desea realizar una copia de seguridad mensual. Introduzca 1 si desea realizar una copia de seguridad mensual.
- Every X MB: This option is used to program backups based on the size of the current request log file. A backup is automatically triggered when the file reaches the set size. Puede definir un límite de tamaño de 1, 10, 100 o 1000 MB.

# Objeto servidor web

Un proyecto 4D puede iniciar y monitorear un servidor web para la aplicación principal (host) así como para cada componente alojado.

Por ejemplo, si ha instalado dos componentes en su aplicación principal, puede iniciar y supervisar hasta tres servidores web independientes desde su aplicación:

- un servidor web para la aplicación local,
- un servidor web para el componente #1,
- un servidor web para el componente #2.

A parte de la memoria, no hay límite en el número de componentes y por lo tanto, de servidores web, que se pueden adjuntar a un solo proyecto de aplicación 4D.

Cada servidor web 4D, incluido el servidor web de la aplicación principal, se expone como un objeto de la clase `4D.WebServer`. Una vez instanciado, un objeto servidor web puede ser manejado desde la aplicación actual o desde cualquier componente utilizando un [gran número de propiedades y funciones](#).

Los [comandos WEB](#) heredados del lenguaje 4D son soportados, pero no se puede seleccionar el servidor web al que se aplican (ver más abajo).

Cada servidor web (aplicación local o componente) puede ser utilizado en su propio contexto independiente, incluyendo:

- las llamadas a los métodos base `On Web Authentication` y `On Web Connection`
- el procesamiento de las etiquetas 4D y las llamadas de métodos,
- sesiones web y gestión del protocolo TLS.

Esto le permite desarrollar componentes independientes y funcionalidades que vienen con sus propias interfaces web.

## Instanciar un objeto servidor web

El objeto servidor web de la aplicación local (servidor web por defecto) es cargado automáticamente por 4D en al inicio. Por lo tanto, si escribe en un proyecto recién creado:

```
$nbSrv:=WEB Server list.length  
//el valor de $nbSrv es 1
```

Para instanciar un objeto servidor web, llame al comando [WEB Server](#):

```
//crear una variable objeto de la clase 4D.WebServer  
var webServer : 4D.WebServer  
    //llamar al servidor web desde el contexto actual  
webServer:=WEB Server  
  
    //equivalente a  
webServer:=WEB Server(Web server database)
```

Si la aplicación utiliza componentes y quiere llamar a:

- el servidor web de la aplicación local a partir de un componente o
- el servidor que ha recibido la solicitud (sin importar el servidor),

también se puede utilizar:

```

var webServer : 4D.WebServer
    //llamar al servidor web local desde un componente
webServer:=WEB Server(Web server host database)
    //llamar al servidor web objetivo
webServer:=WEB Server(Web server receiving request)

```

## Funciones del servidor web

Un [objeto de clase Web server](#) contiene las siguientes funciones:

Funciones	Parámetros	Valor devuelto	Descripción
<code>start()</code>	settings (objet)	status (objeto)	Iniciar el servidor web
<code>stop()</code>	-	-	Detener el servidor Web

Para iniciar y detener un servidor web, basta con llamar a las funciones `start()` y `stop()` del objeto servidor web:

```

var $status : Object
    //para iniciar un servidor web con los parámetros por defecto
$status:=webServer.start()
    //para iniciar el servidor web con los parámetros personalizados
    //settings object contains web server properties
webServer.start($settings)

    //para detener el servidor web
$status:=webServer.stop()

```

## Propiedades del servidor web

Un objeto servidor web contiene [varias propiedades](#) que configuran el servidor web.

Estas propiedades son definidas:

1. con la ayuda del parámetro `settings` de la función `.start()` (excepto en el caso de las propiedades de sólo lectura, ver más adelante),
  2. si no se utiliza, utilizando el comando `WEB SET OPTION` (sólo aplicaciones locales),
  3. si no se utiliza, en los parámetros de la aplicación local o del componente.
- Si el servidor web no se inicia, las propiedades contienen los valores que se utilizarán en el próximo inicio del servidor web.
  - Si el servidor web se inicia, las propiedades contienen los valores reales utilizados por el servidor web (la configuración por defecto puede haber sido reemplazada por el parámetro `settings` de la función `.start()`).

*isRunning, name, openSSLVersion y perfectForwardSecrecy* son propiedades de sólo lectura que no pueden predefinirse en el parámetro del objeto `settings` para la función `start()`.

## Alcance de los comandos 4D Web

El lenguaje 4D contiene [varios comandos](#) permitiendo controlar el servicio Web. Sin embargo, estos comandos están diseñados para trabajar con un único servidor web (por defecto). Cuando utilice estos comandos en el contexto de los objetos servidor web, asegúrese de que su alcance es el adecuado.

Comando	Alcance
SET DATABASE PARAMETER	Aplicación local del servidor web
WEB CLOSE SESSION	Servidor web que ha recibido la petición
WEB GET BODY PART	Servidor web que ha recibido la petición
WEB Get body part count	Servidor web que ha recibido la petición
WEB Get Current Session ID	Servidor web que ha recibido la petición
WEB GET HTTP BODY	Servidor web que ha recibido la petición
WEB GET HTTP HEADER	Servidor web que ha recibido la petición
WEB GET OPTION	Aplicación local del servidor web
WEB Get server info	Aplicación local del servidor web
WEB GET SESSION EXPIRATION	Servidor web que ha recibido la petición
WEB Get session process count	Servidor web que ha recibido la petición
WEB GET STATISTICS	Aplicación local del servidor web
WEB GET VARIABLES	Servidor web que ha recibido la petición
WEB Is secured connection	Servidor web que ha recibido la petición
WEB Is server running	Aplicación local del servidor web
WEB SEND BLOB	Servidor web que ha recibido la petición
WEB SEND FILE	Servidor web que ha recibido la petición
WEB SEND HTTP REDIRECT	Servidor web que ha recibido la petición
WEB SEND RAW DATA	Servidor web que ha recibido la petición
WEB SEND TEXT	Servidor web que ha recibido la petición
WEB SET HOME PAGE	Aplicación local del servidor web
WEB SET HTTP HEADER	Servidor web que ha recibido la petición
WEB SET OPTION	Aplicación local del servidor web
WEB SET ROOT FOLDER	Aplicación local del servidor web
WEB START SERVER	Aplicación local del servidor web
WEB STOP SERVER	Aplicación local del servidor web
WEB Validate digest	Servidor web que ha recibido la petición

# Páginas de plantillas

El servidor web de 4D le permite utilizar las páginas de plantillas HTML que contengan etiquetas, es decir, una mezcla de código HTML estático y de referencias 4D añadidas mediante [etiquetas de transformación](#) como 4DTEXT, 4DIF o 4DINCLUDE. Estas etiquetas suelen insertarse como comentarios de tipo HTML (`<!--#4DTagName TagValue-->`) en el código fuente HTML.

Cuando estas páginas son enviadas por el servidor HTTP, son analizadas y las etiquetas que contienen son ejecutadas y sustituidas por los datos resultantes. Las páginas que reciben los navegadores son una combinación de elementos estáticos y valores procedentes del procesamiento 4D.

Por ejemplo, si se escribe en una página HTML:

```
<P>Welcome to <!--#4DTEXT vtSiteName-->!</P>
```

El valor de la variable 4D `vtSiteName` se insertará en la página HTML.

## Etiquetas para las plantillas

Las siguientes etiquetas 4D están disponibles:

- 4DTEXT, para insertar variables y expresiones 4D como texto,
- 4DHTML, para insertar el código HTML,
- 4DEVAL, para evaluar toda expresión 4D,
- 4DSCRIPT, para ejecutar un método 4D,
- 4DINCLUDE, para incluir una página dentro de otra,
- 4DBASE, para modificar la carpeta por defecto utilizada por la etiqueta 4DINCLUDE,
- 4DCODE, para insertar el código 4D,
- 4DIF, 4ELSE, 4ELSEIF y 4ENDIF, para insertar condiciones en el código HTML,
- 4DLOOP y 4DENDLOOP, para hacer bucles en el código HTML,
- 4DEACH y 4DENDEACH, para hacer bucles en colecciones, selecciones de entidades o propiedades de objetos.

Estas etiquetas se describen en la página [Etiquetas de transformación](#).

Es posible combinar etiquetas. Por ejemplo, el siguiente código HTML es permitido:

```

<HTML>
...
<BODY>
<!--#4DSCRIPT/PRE_PROCESS--> (Method call)
<!--#4DIF (myvar=1)--> (If condition)
    <!--#4DINCLUDE banner1.html--> (Subpage insertion)
<!--#4DENDIF--> (End if)
<!--#4DIF (myvar=2)-->

    <!--#4DINCLUDE banner2.html-->
<!--#4DENDIF-->

<!--#4DLOOP [TABLE]--> (loop on the current selection)
<!--#4DIF ([TABLE]ValNum>10)--> (If [TABLE]ValNum>10)
    <!--#4DINCLUDE subpage.html--> (subpage insertion)
<!--#4DELSE--> (Else)
    <B>Value: <!--#4DTEXT [TABLE]ValNum--></B><BR>
        (Field display)
<!--#4DENDIF-->
<!--#4DENDLOOP--> (End for)
</BODY>
</HTML>

```

## Análisis de etiquetas

Por razones de optimización, el servidor web de 4D no realiza el análisis del código fuente HTML cuando se llama a las páginas HTML utilizando URLs simples con el sufijo ".HTML" o ".HTM".

El análisis del contenido de las páginas plantillas enviadas por el servidor web de 4D tiene lugar cuando se llaman los comandos `WEB SEND FILE` (.htm, .html, .shtm, .shtml), `WEB SEND BLOB` (text/html type BLOB) o `WEB SEND TEXT`, así como cuando se envían páginas llamadas mediante URLs. En este último caso, por razones de optimización, las páginas con sufijo ".htm" y ".html" NO se analizan. Para "forzar" el análisis de las páginas HTML en este caso, debe añadir el sufijo ".shtm" o ".shtml" (por ejemplo, <http://www.server.com/dir/page.shtm>). Un ejemplo del uso de este tipo de página se da en la descripción del comando `WEB GET STATISTICS`. Las páginas XML (.xml, .xsl) también son compatibles y siempre son analizadas por 4D.

También puede llevar a cabo el análisis sintáctico fuera del contexto web cuando utilice el comando `PROCESS 4D TAGS`.

Internamente, el analizador funciona con cadenas UTF-16, pero los datos a analizar pueden haber sido codificados de forma diferente. Cuando las etiquetas contienen texto (por ejemplo `4DHTML`), 4D convierte los datos cuando es necesario dependiendo de su origen y de la información disponible (charset). A continuación se muestran los casos en los que 4D analiza las etiquetas contenidas en las páginas HTML, así como las conversiones realizadas:

Acción / Comando	Análisis del contenido de las páginas enviadas	Soporte de la sintaxis \$(*)	Conjunto de caracteres utilizados para el análisis sintáctico de las etiquetas
Páginas llamadas a través de URLs	X, excepto las páginas con extensiones ".htm" o ".html"	X, excepto las páginas con extensiones ".htm" o ".html"	Uso del conjunto de caracteres pasado como parámetro del encabezado "Content-Type" de la página. Si no hay ninguna, busca una etiqueta META-HTTP EQUIV con un conjunto de caracteres. En caso contrario, uso del conjunto de caracteres por defecto para el servidor HTTP
WEB SEND FILE	X	-	Uso del conjunto de caracteres pasado como parámetro del encabezado "Content-Type" de la página. Si no hay ninguna, busca una etiqueta META-HTTP EQUIV con un conjunto de caracteres. En caso contrario, uso del conjunto de caracteres por defecto para el servidor HTTP
WEB SEND TEXT	X	-	No es necesaria la conversión
WEB SEND BLOB	X, si el BLOB es de tipo "text/html"	-	Uso del conjunto de caracteres definido en el encabezado "Content-Type" de la respuesta. En caso contrario, uso del conjunto de caracteres por defecto para el servidor HTTP
Inclusión por la etiqueta <!-- #4DINCLUDE-->	X	X	Uso del conjunto de caracteres pasado como parámetro del encabezado "Content-Type" de la página. Si no hay ninguna, busca una etiqueta META-HTTP EQUIV con un conjunto de caracteres. En caso contrario, uso del conjunto de caracteres por defecto para el servidor HTTP
PROCESS 4D TAGS	X	X	Datos de texto: no hay conversión. Datos BLOB: conversión automática del conjunto de caracteres Mac-Roman por compatibilidad

(\*) La sintaxis alternativa basada en \$ está disponible para las etiquetas 4DHTML, 4DTEXT y 4DEVAL.

## Acceso a los métodos 4D a través de la web

La ejecución de un método 4D con 4DEACH , 4DELSEIF , 4DEVAL , 4DHTML , 4DIF , 4DL0OP , 4DSCRIPT , o 4DTEXT a partir de una petición web está sujeta al valor del atributo [disponible vía las etiquetas 4D y las URL \(4ACTION...\)](#) definida en las propiedades del método. Si no se comprueba el atributo para el método, éste no puede ser llamado desde una petición web.

## Prevención de la inserción de códigos maliciosos

Las etiquetas 4D aceptan diferentes tipos de datos como parámetros: texto, variables, métodos, nombres de comandos, etc. Cuando estos datos son proporcionados por su propio código, no hay riesgo de inserción de código malicioso ya que usted controla la entrada. Sin embargo, el código de su base de datos suele trabajar con datos que, en un momento u otro, fueron introducidos a través de una fuente externa (entrada del usuario, importación, etc.).

En este caso, es aconsejable no utilizar etiquetas como 4DEVAL o 4DSCRIPT , que evalúan parámetros, directamente con este tipo de datos.

Además, según el [principio de recursividad](#), el código malicioso puede incluir a su vez etiquetas de transformación. En este caso, es imprescindible utilizar la etiqueta 4DTEXT . Imagine, por ejemplo, un campo de formulario web llamado "Name", donde los usuarios deben introducir su nombre. Este nombre se muestra mediante una etiqueta <!--#4DHTML vName--> en la página. Si se inserta un texto del tipo "<!--#4DEVAL QUIT 4D-->" en lugar del nombre, la interpretación de esta etiqueta provocará la salida de la aplicación. Para evitar este riesgo, basta con utilizar

sistématicamente la etiqueta `<4DTEXT>` en este caso. Como esta etiqueta escapa a los caracteres especiales de HTML, cualquier código recursivo malicioso que pueda haberse insertado no será reinterpretado. Para referirnos al ejemplo anterior, el campo "Name" contendrá, en este caso, "`< &lt;!--#4DEVAL QUIT 4D-->`" que no será transformado.

# Procesamiento de peticiones HTTP

El servidor web de 4D ofrece varias funcionalidades para gestionar las peticiones HTTP:

- el método base `On Web Connection`, un enrutador para su aplicación web,
- la URL `/4DACTION` para llamar al código del lado del servidor
- `WEB GET VARIABLES` para obtener valores de los objetos HTML enviados al servidor
- otros comandos como `WEB GET HTTP BODY`, `WEB GET HTTP HEADER`, o `WEB GET BODY PART` permiten personalizar el tratamiento de las solicitudes, incluidas las cookies.
- el método proyecto `COMPILER_WEB`, para declarar sus variables.

## On Web Connection

El método base `On Web Connection` puede utilizarse como punto de entrada al servidor web de 4D.

### Llamadas a métodos base

El método base `On Web Connection` se llama automáticamente cuando el servidor recibe cualquier URL que no sea una ruta a una página existente en el servidor. Se llama al método de la base de datos con la URL.

Por ejemplo, la URL "a/b/c" llamará al método base, pero "a/b/c.html" no llamará al método base si la página "c.html" existe en la subcarpeta "a/b" del [WebFolder](#).

La petición debe haber sido aceptada previamente por el método base `On Web Authentication` (si existe) y el servidor web debe ser lanzado.

## Sintaxis

`On Web Connection( $1 : Text ; $2 : Text ; $3 : Text ; $4 : Text ; $5 : Text ; $6 : Text )`

Parámetros	Tipo		Descripción
\$1	Texto	<-	URL
\$2	Texto	<-	Encabezados HTTP + cuerpo HTTP (hasta un límite de 32 kb)
\$3	Texto	<-	Dirección IP del cliente web (navegador)
\$4	Texto	<-	Dirección IP del servidor
\$5	Texto	<-	Nombre de usuario
\$6	Texto	<-	Contraseña

Debe declarar estos parámetros de la siguiente manera:

```
//Método base On Web Connection  
  
C_TEXT($1;$2;$3;$4;$5;$6)  
  
//Código para el métodod
```

Como alternativa, puede utilizar la sintaxis [parámetros nombrados](#):

```
// Método base On Web Connection
#DECLARE ($url : Text; $header : Text; \
$BrowserIP : Text; $ServerIP : Text; \
$user : Text; $password : Text)
```

Llamar a un comando 4D que muestra un elemento de interfaz ( `DIALOG` , `ALERT` , etc.) no está permitido y termina el procesamiento del método.

## \$1 - Datos adicionales de la URL

El primer parámetro (`$1`) es la URL introducida por los usuarios en el área de direcciones de su navegador web, sin la dirección local.

Utilicemos como ejemplo una conexión de intranet. Supongamos que la dirección IP de su máquina 4D Web Server es 123.4.567.89. La siguiente tabla muestra los valores de `$1` en función de la URL introducida en el navegador web:

URL introducida en el navegador web	Valor del parámetro <code>\$1</code>
123.4.567.89	/
<a href="http://123.4.567.89">http://123.4.567.89</a>	/
123.4.567.89/Customers	/Customers
<a href="http://123.4.567.89/Customers/Add">http://123.4.567.89/Customers/Add</a>	/Customers/Add
123.4.567.89/Do_This/If_OK/Do_That	/Do_This/If_OK/Do_That

Tenga en cuenta que es libre de utilizar este parámetro a su conveniencia. 4D simplemente ignora el valor pasado más allá de la parte del host de la URL. Por ejemplo, puede establecer una convención en la que el valor `"/Customers/Add"` significa "ir directamente a añadir un nuevo registro en la tabla `[Customers]` ". Al proporcionar a los usuarios de la web una lista de posibles valores y/o marcadores por defecto, puede dar accesos directos a diferentes partes de su aplicación. De este modo, los usuarios de la web pueden acceder rápidamente a los recursos de su sitio web sin tener que recorrer toda la ruta de navegación cada vez que realicen una nueva conexión.

## \$2 - Encabezado y cuerpo de la petición HTTP

El segundo parámetro (`$2`) es el encabezado y el cuerpo de la petición HTTP enviada por el navegador web. Tenga en cuenta que esta información se pasa a su método base `On Web Connection` "tal cual". Su contenido variará en función de la naturaleza del navegador web que intenta la conexión.

Si su aplicación utiliza esta información, deberá analizar el encabezado y el cuerpo. Puede utilizar los comandos `WEB GET HTTP HEADER` y `WEB GET HTTP BODY` .

Por razones de rendimiento, el tamaño de los datos que pasan por el parámetro `$2` no debe superar los 32 KB. Más allá de este tamaño, son truncados por el servidor HTTP de 4D.

## \$3 - Dirección IP del cliente web

El parámetro `$3` recibe la dirección IP de la máquina del navegador. Esta información puede permitirle distinguir entre las conexiones a la intranet y a Internet.

4D devuelve las direcciones IPv4 en un formato híbrido IPv6/IPv4 escrito con un prefijo de 96 bits, por ejemplo `::ffff:192.168.2.34` para la dirección IPv4 192.168.2.34. Para más información, consulte la sección [Soporte IPv6](#).

## \$4 - Dirección IP del servidor

El parámetro \$4 recibe la dirección IP solicitada por el servidor web 4D. 4D permite el multi-homing, que permite utilizar máquinas con más de una dirección IP. Para más información, consulte la [página Configuración](#).

## \$5 y \$6 - Nombre de usuario y contraseña

Los parámetros \$5 y \$6 reciben el nombre de usuario y la contraseña introducidos por el usuario en el cuadro de diálogo de identificación estándar que muestra el navegador, si procede (ver la página [autenticación](#)).

Si el nombre de usuario enviado por el navegador existe en 4D, el parámetro \$6 (la contraseña del usuario) no se devuelve por razones de seguridad.

## /4DACTION

/4DACTION/MethodName\*\*\*

\*/4DACTION/\*\*\*\*\*MethodName/Param

Parámetros	Tipo		Descripción
MethodName	Texto	->	Nombre del método de proyecto 4D a ejecutar
Param	Texto	->	Parámetro texto a pasar al método proyecto

Uso: URL o acción del formulario.

Esta URL permite llamar al método proyecto 4D *MethodName* con un parámetro texto opcional *Param*. El método recibirá este parámetro en \$1.

- El método proyecto 4D debe haber sido [permitido para peticiones web](#): el valor del atributo "Disponible a través de etiquetas y URLs 4D (4DACTION...)" debe haber sido marcado en las propiedades del método. Si no se comprueba el atributo, se rechaza la solicitud web.
- Cuando 4D recibe una petición /4DACTION/MethodName/Param , se llama el método base On Web Authentication (si existe).

4DACTION/ puede asociarse a una URL en una página web estática:

```
<A HREF="/4DACTION/MyMethod/hello">Do Something</A>
```

El método proyecto MyMethod debe devolver generalmente una "respuesta" (envío de una página HTML utilizando WEB SEND FILE o WEB SEND TEXT , etc.). mediante. Asegúrese de que el tratamiento sea lo más breve posible para no bloquear el navegador.

Un método llamado por /4DACTION no debe llamar a elementos de la interfaz ( DIALOG , ALERT , etc.).

### Ejemplo

Este ejemplo describe la asociación de la URL /4DACTION con un objeto imagen HTML para mostrar dinámicamente una imagen en la página. Inserta las siguientes instrucciones en una página HTML estática:

```
<IMG SRC="/4DACTION/getPhoto/smith">
```

El método getPhoto es el siguiente:

```

C_TEXT($1) // Este parámetro debe declararse siempre
var $path : Text
var $PictVar : Picture
var $BlobVar : Blob

//busca la imagen en la carpeta Imágenes dentro de la carpeta Resources
$path:=Get 4D folder(Current resources folder)+"Images"+Folder separator+$1+".psd"

READ PICTURE FILE($path;$PictVar) //pone la imagen en la variable imagen
PICTURE TO BLOB($PictVar;$BLOB;".png") //convierte la imagen en formato ".png".
WEB SEND BLOB($BLOB;"image/png")

```

## 4DACCIÓN para publicar formularios

El servidor web de 4D también le permite utilizar formularios "publicados", que son páginas HTML estáticas que envían datos al servidor web y recuperar fácilmente todos los valores. Se les debe asociar el tipo POST y la acción del formulario debe empezar imperativamente por /4DACTION/MethodName.

Un formulario puede ser enviado a través de dos métodos (ambos pueden ser utilizados con 4D):

- POST, normalmente utilizado para enviar datos al servidor web,
- GET, normalmente utilizado para solicitar datos del servidor web.

Cuando el servidor web recibe un formulario publicado, llama al método base `On Web Authentication` (si existe).

En el método llamado, debe llamar al comando `WEB GET VARIABLES` para [recuperar los nombres y valores](#) de todos los campos incluidos en una página HTML enviada al servidor.

Ejemplo para definir la acción de un formulario:

```
<FORM ACTION="/4DACTION/MethodName" METHOD=POST>
```

### Ejemplo

En una aplicación web, nos gustaría que los navegadores pudieran buscar entre los registros mediante una página HTML estática. Esta página se llama "search.htm". La aplicación contiene otras páginas estáticas que permiten, por ejemplo, mostrar el resultado de la búsqueda ("results.htm"). Se ha asociado el tipo POST a la página, así como la acción `/4DACTION/SEARCH`.

Aquí está el código HTML que corresponde a esta página:

```

<form action="/4daction/processForm" method=POST>
<input type=text name=vName value=""><BR>
<input type=checkbox name=vExact value="Word">Whole word<BR>
<input type=submit name=OK value="Search">
</FORM>

```

Durante la entrada de datos, escriba "ABCD" en el área de entrada de datos, marque la opción "Whole word" y valídela haciendo clic en el botón Search. En la solicitud enviada al servidor web:

```

vName="ABCD"
vExact="Word"
OK="Search"

```

4D llama al método base `<On Web Authentication>` (si existe), luego se llama al método proyecto `processForm`, que es el siguiente:

```

C_TEXT($1) //obligatorio para el modo compilado
C_LONGINT($vName)
C_TEXT(vName;vLIST)
ARRAY TEXT($arrNames;0)
ARRAY TEXT($arrVals;0)
WEB GET VARIABLES($arrNames;$arrVals) //recuperamos todas las variables del formulario
$vName:=Find in array($arrNames;"vName")
vName:=$arrVals{$vName}
If(Find in array($arrNames;"vExact")=-1) //Si la opción no ha sido marcada
    vName:=vName+"@"
End if
QUERY([Jockeys];[Jockeys]Name=vName)
FIRST RECORD([Jockeys])
While(Not(End selection([Jockeys])))
    vLIST:=vLIST+[Jockeys]Name+" "+[Jockeys]Tel+"  
"
    NEXT RECORD([Jockeys])
End while
WEB SEND FILE("results.htm") //Enviar la lista al formulario results.htm
//que contiene una referencia a la variable vLIST,
//por ejemplo <!--4DHTML vLIST-->
//...
End if

```

## Obtener valores de las peticiones HTTP

El servidor web de 4D le permite recuperar datos enviados a través de peticiones POST o GET, utilizando formularios web o URLs.

Cuando el servidor web recibe una petición con datos en el encabezado o en la URL, 4D puede recuperar los valores de cualquier objeto HTML que contenga. Este principio puede implementarse en el caso de un formulario web, enviado por ejemplo utilizando `WEB SEND FILE` o `WEB SEND BLOB`, en el que el usuario introduce o modifica valores y luego hace clic en el botón de validación.

En este caso, 4D puede recuperar los valores de los objetos HTML encontrados en la petición utilizando el comando `WEB GET VARIABLES`. El comando `WEB GET VARIABLES` recupera los valores como texto.

Considere el siguiente código fuente de la página HTML:

```

<html>
<head>
    <title>Welcome</title>
    <script language="JavaScript"><!--
function GetBrowserInformation(formObj){
formObj.vtNav_appName.value = navigator.appName
formObj.vtNav_appVersion.value = navigator.appVersion
formObj.vtNav_appCodeName.value = navigator.appCodeName
formObj.vtNav_userAgent.value = navigator.userAgent
return true
}
function LogOn(formObj){
if(formObj.vtUserName.value!=""){
return true
} else {
alert("Enter your name, then try again.")
return false
}
}
//--></script>
</head>
<body>
<form action="/4DACTION/WWW_STD_FORM_POST" method="post"
name="frmWelcome"
onsubmit="return GetBrowserInformation(frmWelcome)">
    <h1>Welcome to Spiders United</h1>
    <p><b>Please enter your name:</b>
    <input name="vtUserName" value="" size="30" type="text"></p>
    <p>
<input name="vsbLogOn" value="Log On" onclick="return LogOn(frmWelcome)" type="submit">
<input name="vsbRegister" value="Register" type="submit">
<input name="vsbInformation" value="Information" type="submit"></p>
<p>
<input name="vtNav_appName" value="" type="hidden">
<input name="vtNav_appVersion" value="" type="hidden">
<input name="vtNav_appCodeName" value="" type="hidden">
<input name="vtNav_userAgent" value="" type="hidden"></p>
</form>
</body>
</html>

```

Cuando 4D envía la página a un navegador web, tiene el siguiente aspecto:

Welcome to Spiders United

Please enter your name:

Las principales características de esta página son:

- Incluye tres botones Submit: `vsbLogOn`, `vsbRegister` y `vsbInformation`.
- Cuando se hace clic en Log On, el envío del formulario se procesa primero por la función de JavaScript `LogOn`. Si no se introduce ningún nombre, el formulario ni siquiera se envía a 4D, y se muestra una alerta de JavaScript.
- El formulario tiene un método POST 4D así como un script Submit (`GetBrowserInformation`) que copia las propiedades del navegador a los cuatro objetos ocultos cuyos nombres empiezan por `vtNav_App`. También incluye el objeto `vtUserName`.

Examinemos el método 4D `WWW_STD_FORM_POST` que se llama cuando el usuario hace clic en uno de los botones del formulario HTML.

```

// Recuperación del valor de las variables
ARRAY TEXT($arrNames;0)
ARRAY TEXT($arrValues;0)
WEB GET VARIABLES($arrNames;$arrValues)
C_TEXT($user)

Case of

// Se ha presionado el botón Log On
:(Find in array($arrNames;"vsbLogOn")#-1)
$user :=Find in array($arrNames;"vtUserName")
QUERY([WWW Users];[WWW Users]UserName=$arrValues{$user})
$0:=(Records in selection([WWW Users])>0)
If($0)
    WWW POST EVENT("Log On";WWW Log information)
// El método WWW POST EVENT guarda la información en una tabla de la base
Else

    $0:=WWW Register
// El método WWW Register permite que un nuevo usuario de la Web se registre
End if

// Se ha presionado el botón Register
:(Find in array($arrNames;"vsbRegister")#-1)
$0:=WWW Register

// Se ha presionado el botón de información
:(Find in array($arrNames;"vsbInformation")#-1)
WEB SEND FILE("userinfos.html")
End case

```

Las funcionalidades de este método son:

- Los valores de las variables *vtNav\_appName*, *vtNav\_appVersion*, *vtNav\_appCodeName*, y *vtNav\_userAgent* (vinculadas a los objetos HTML que tienen los mismos nombres) se recuperan utilizando el comando `WEB GET VARIABLES` a partir de los objetos HTML creados por el script JavaScript *GetBrowserInformation*.
- De las variables vinculadas *vsbLogOn*, *vsbRegister* y *vsbInformation* a los tres botones de envío, sólo la correspondiente al botón que se ha presionado será recuperada por el comando `WEB GET VARIABLES`. Cuando el envío se realiza mediante uno de estos botones, el navegador devuelve a 4D el valor del botón presionado. Esto le indica qué botón se ha presionado.

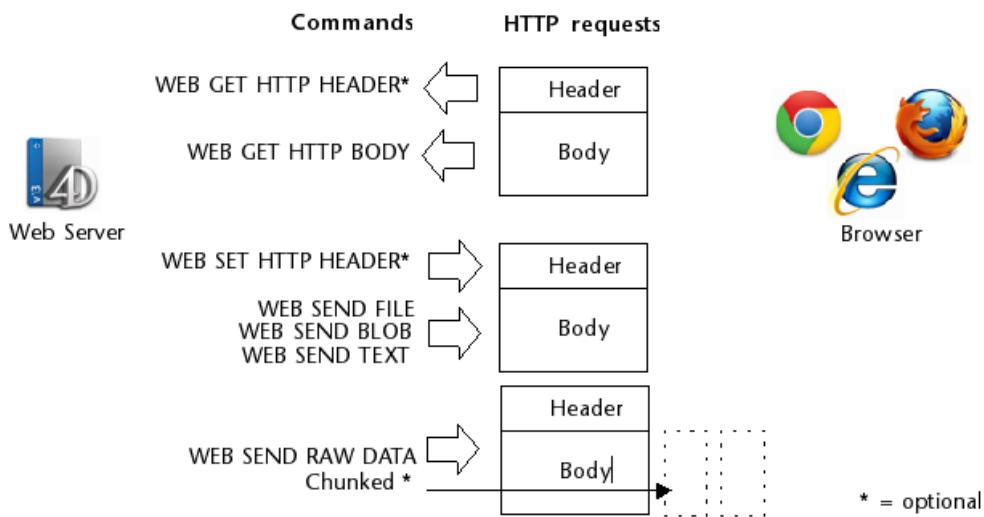
Tenga en cuenta que con HTML, todos los objetos son objetos de texto. Si se utiliza un objeto SELECT, es el valor del elemento resaltado en el objeto el que se devuelve en el comando `WEB GET VARIABLES`, y no la posición del elemento en el array como en 4D. `WEB GET VARIABLES` siempre devuelve valores de tipo Texto.

## Otros comandos del servidor web

El servidor web de 4D ofrece varios comandos web de bajo nivel que le permiten desarrollar un procesamiento personalizado de las solicitudes:

- el comando `WEB GET HTTP BODY` devuelve el cuerpo como texto sin procesar, permitiendo cualquier análisis que pueda necesitar
- el comando `WEB GET HTTP HEADER` devuelve los encabezados de la petición. Es útil para manejar cookies personalizadas, por ejemplo (junto con el comando `WEB SET HTTP HEADER`).
- los comandos `WEB GET BODY PART` y `WEB Get body part count` para analizar la parte del cuerpo de una petición de varias partes y recuperar los valores de texto, pero también los archivos publicados, utilizando BLOBs.

Estos comandos se resumen en el siguiente gráfico:



El servidor web de 4D ahora soporta archivos cargados con codificación chunked desde cualquier cliente web. La codificación de transferencia en trozos es un mecanismo de transferencia de datos especificado en HTTP/1.1. Permite transferir datos en una serie de "trozos" (partes) sin conocer el tamaño final de los datos. El servidor web de 4D también soporta la codificación de transferencia en trozos desde el servidor a los clientes web (utilizando `WEB SEND RAW DATA` ).

## Método proyecto COMPILER\_WEB

El método `COMPILERWEB`, si existe, es llamado sistemáticamente cuando el servidor HTTP recibe una petición dinámica y llama al motor 4D. Este es el caso, por ejemplo, cuando el servidor web de 4D recibe un formulario publicado o una URL para procesar en [On Web Connection](#). Este método está destinado a contener directivas de digitación y/o inicialización de variables utilizadas durante los intercambios web. Es utilizado por el compilador cuando se compila la aplicación. El método `COMPILERWEB` es común a todos los formularios web. Por defecto, el método `COMPILER_WEB` no existe. Debe crearlo explícitamente.

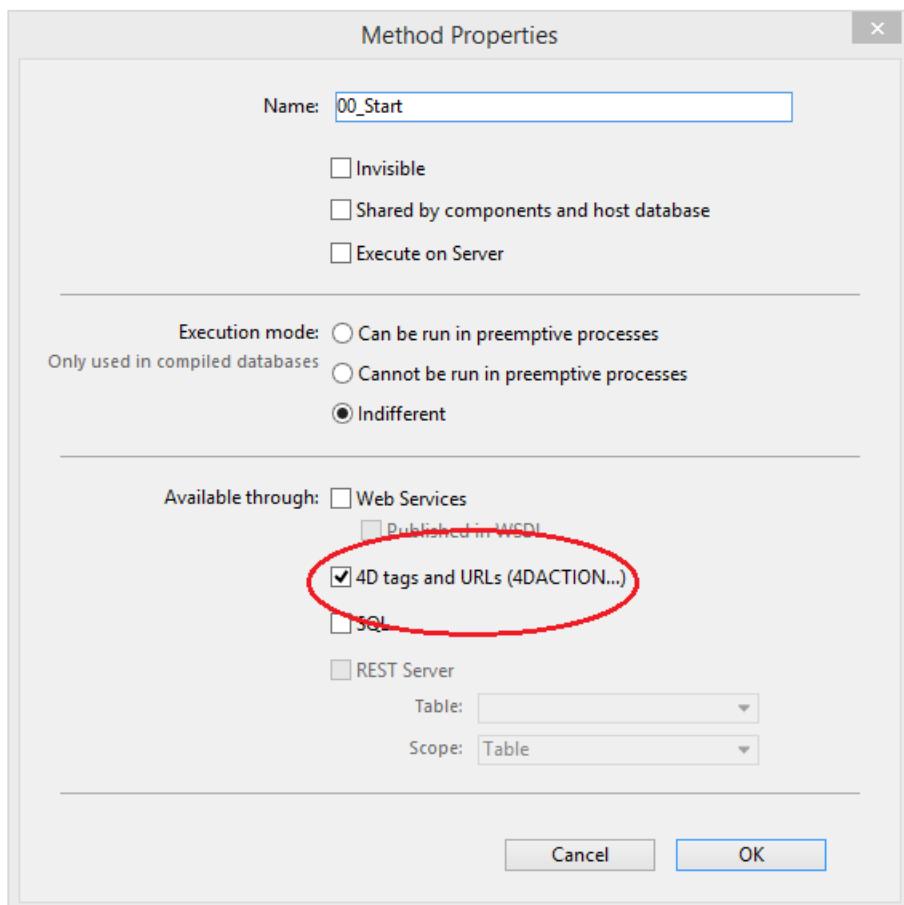
También se llama al método proyecto `COMPILER_WEB`, si existe, para cada solicitud SOAP aceptada.

# Permitir métodos proyecto

Las etiquetas 4D como `4DEVAL`, `4DTEXT`, `4DHTML`... así como la URL `/4ACTION` permiten desencadenar la ejecución de cualquier método de un proyecto 4D publicado en la web. Por ejemplo, la petición <http://www.server.com/4ACTION/login> provoca la ejecución del método proyecto `login`, si existe.

Por lo tanto, este mecanismo presenta un riesgo de seguridad para la aplicación, en particular si un usuario de Internet activa intencionalmente (o no) un método no previsto para su ejecución a través de la web. Puede evitar este riesgo de las siguientes maneras:

- Filtre los métodos llamados a través de las URLs utilizando el método base [On Web Authentication](#).  
Inconvenientes: si la base de datos incluye un gran número de métodos, este sistema puede ser difícil de gestionar.
- Utilice la opción Disponible a través de etiquetas 4D y URLs (4ACTION...) que se encuentra en la caja de diálogo de propiedades del método:



Esta opción se utiliza para designar individualmente cada método del proyecto que puede ser llamado utilizando la URL especial `4ACTION`, o las etiquetas `4DTEXT`, `4DHTML`, `4DEVAL`, `4SCRIPT`, `4DIF`, `4ELSEIF` o `4LOOP`. Cuando no está marcada, el método proyecto en cuestión no puede ser ejecutado directamente a través de una petición HTTP. Por el contrario, puede ejecutarse mediante otro tipo de llamadas (fórmulas, otros métodos, etc.).

Esta opción está deselegionada por defecto. Los métodos que se pueden ejecutar a través de `4ACTION` o de etiquetas específicas deben indicarse específicamente.

En el Explorador, los métodos proyecto con esta propiedad reciben un icono específico:



# Páginas de error HTTP personalizadas

El servidor web de 4D le permite personalizar las páginas de error HTTP enviadas a los clientes, basándose en el código de estado de la respuesta del servidor. Las páginas de error se refieren a:

- los códigos de estado que empiezan por 4 (errores del cliente), por ejemplo 404
- los códigos de estado que empiezan por 5 (errores del servidor), por ejemplo 501.

Para una descripción completa de los códigos de estado de error HTTP, puede consultar la [lista de códigos de estado HTTP](#) (Wikipedia).

## Reemplazo de las páginas por defecto

Para reemplazar las páginas de error predeterminadas de 4D Web Server por sus propias páginas, sólo tiene que:

- poner las páginas HTML personalizadas en el primer nivel de la carpeta web de la aplicación,
- nombrar las páginas personalizadas "{statusCode}.html" (por ejemplo, "404.html").

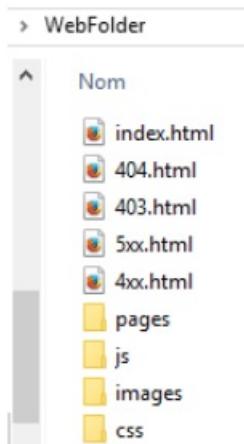
Puede definir una página de error por código de estado y/o una página de error genérica para un rango de errores, llamada "{number}xx.html". Por ejemplo, puede crear "4xx.html" para los errores genéricos del cliente. El servidor web de 4D buscará primero una página {statusCode}.html y, si no existe, una página genérica.

Por ejemplo, cuando una respuesta HTTP devuelve un código de estado 404:

1. 4D Web Server intenta enviar una página "404.html" ubicada en la carpeta web de la aplicación.
2. Si no se encuentra, 4D Web Server intenta enviar una página "404.html" ubicada en la carpeta web de la aplicación.
3. Si no se encuentra, 4D Web Server utiliza su página de error por defecto.

## Ejemplo

Si define las siguientes páginas personalizadas en su carpeta web:



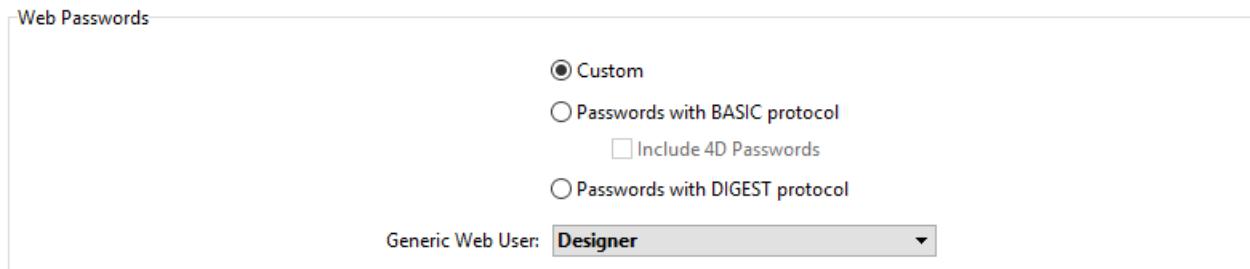
- se servirán las páginas "403.html" o "404.html" cuando se devuelvan respuestas HTTP 403 o 404 respectivamente,
- la página "4xx.html" se servirá para cualquier otro estado de error 4xx (400, 401, etc.),
- la página "5xx.html" se servirá para cualquier estado de error 5xx.

# Autenticación

La autenticación de los usuarios es necesaria cuando se desea ofrecer derechos de acceso específicos a los usuarios Web. La autenticación designa el modo en que se recoge y procesa la información relativa a las credenciales del usuario (normalmente nombre y contraseña).

## Modos de autenticación

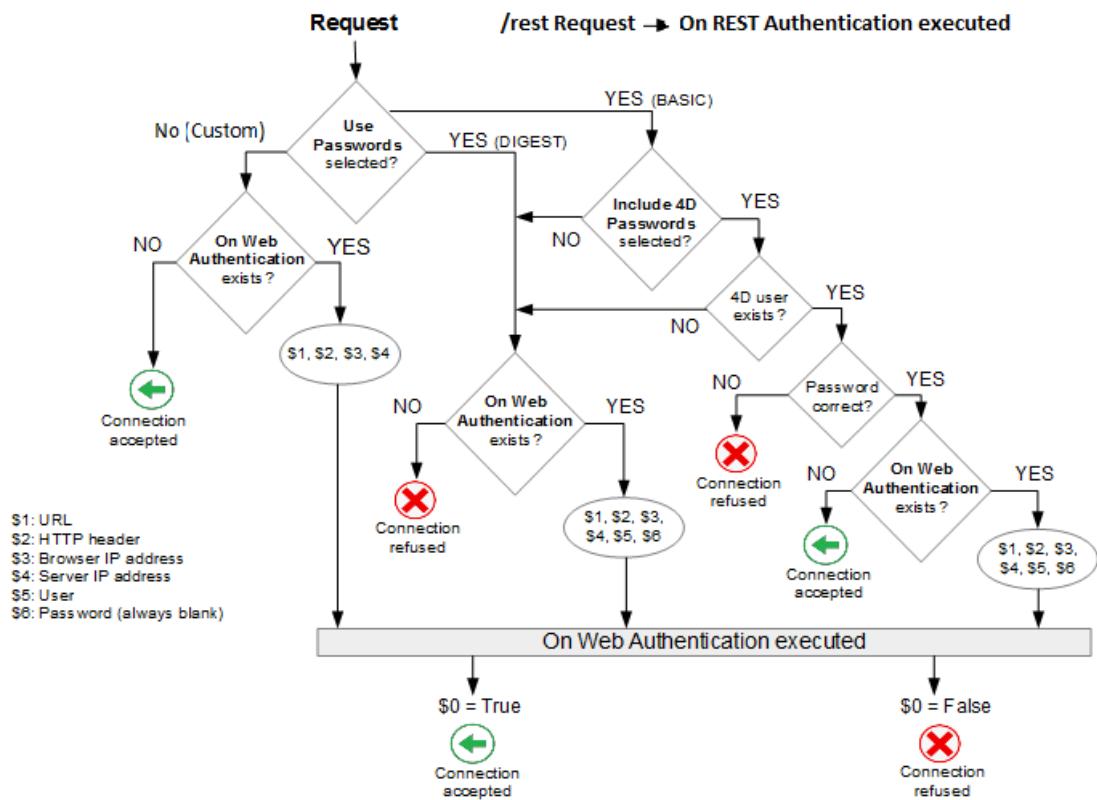
El servidor web 4D ofrece tres modos de autenticación, que puede seleccionar en la página Web/Opciones (I) de la ventana Propiedades:



Se recomienda utilizar una autenticación personalizada.

## Generalidades

El funcionamiento del sistema de acceso del servidor web 4D se resume en el siguiente diagrama:



Las peticiones que comienzan por `rest/` son gestionadas directamente por el [servidor REST](#).

## Personalizado (por defecto)

Básicamente, en este modo, depende del desarrollador definir cómo autenticar a los usuarios. 4D sólo evalúa las peticiones HTTP [que requieren una autenticación](#).

Este modo de autenticación es el más flexible porque permite:

- o bien, delegar la autenticación del usuario a una aplicación de terceros (por ejemplo, una red social, SSO);
- o bien, ofrecer una interfaz al usuario (por ejemplo, un formulario web) para que pueda crear su cuenta en su base de datos clientes; luego, puede autenticar a los usuarios con cualquier algoritmo personalizado (ver [este ejemplo](#) del capítulo "Sesiones Usuario"). Lo importante es que nunca guarde la contraseña en claro, utilizando ese código:

```
//... creación de cuenta de usuario  
ds.webUser.password:=Generate password hash($password)  
ds.webUser.save()
```

Ver también [este ejemplo](#) del capítulo "Cómo comenzar".

Si no se facilita una autenticación personalizada, 4D llama al método de base de datos [On Web Authentication](#) (si existe). Además de \$1 y \$2, sólo se facilitan las direcciones IP del navegador y del servidor (\$3 y \$4), el nombre de usuario y la contraseña (\$5 y \$6) están vacíos. El método debe devolver True en \$0 si el usuario se autentifica con éxito, entonces se sirve el recurso solicitado, o False en \$0 si la autenticación falló.

Atención: si el método de base de datos [On Web Authentication](#) no existe, las conexiones se aceptan automáticamente (modo de prueba).

## Protocolo Basic

Cuando un usuario se conecta al servidor, aparece una caja de diálogo estándar en su navegador para que introduzca su nombre de usuario y contraseña.

El nombre y la contraseña introducidos por el usuario se envían sin cifrar en el encabezado de la petición HTTP. Este modo suele requerir HTTPS para ofrecer confidencialidad.

A continuación, se evalúan los valores introducidos:

- Si la opción Incluir contraseñas de 4D está marcada, las credenciales de los usuarios se evaluarán primero contra la [tabla interna de usuarios 4D](#).
  - Si el nombre de usuario enviado por el navegador existe en la tabla de usuarios 4D y la contraseña es correcta, se acepta la conexión. Si la contraseña es incorrecta, se rechaza la conexión.
  - Si el nombre de usuario no existe en la tabla de usuarios 4D, se llama al método base [On Web Authentication](#). Si el método base [On Web Authentication](#) no existe, se rechazan las conexiones.
- Si la opción Incluir las contraseñas de 4D no está marcada, las credenciales de usuario se envían al método base [On Web Authentication](#) junto con los demás parámetros de conexión (dirección y puerto IP, URL...) para que pueda procesarlas. Si el método base [On Web Authentication](#) no existe, se rechazan las conexiones.

Con el servidor Web del cliente 4D, tenga en cuenta que todos los sitios publicados por las máquinas 4D Client compartirán la misma tabla de usuarios. La validación de los usuarios/contraseñas la realiza la aplicación 4D Server.

## Protocolo DIGEST

Este modo ofrece un mayor nivel de seguridad, ya que la información de autenticación se procesa mediante un proceso unidireccional llamado hashing que hace que su contenido sea imposible de descifrar.

Al igual que en el modo BASIC, los usuarios deben introducir su nombre y contraseña al conectarse. A continuación, se llama al método base [On Web Authentication](#). Cuando se activa el modo DIGEST, el parámetro \$6 (contraseña) se devuelve siempre vacío. De hecho, cuando se utiliza este modo, esta información no pasa por la red como texto claro (sin encriptar). Por lo tanto, en este caso es imprescindible evaluar las solicitudes de conexión mediante el comando

WEB Validate digest .

Debe reiniciar el servidor web para que se tengan en cuenta los cambios realizados en estos parámetros.

## On Web Authentication

El método de base de datos **On Web Authentication** se encarga de gestionar el acceso al motor del servidor web. Es llamado por 4D o 4D Server cuando se recibe una petición HTTP dinámica.

### Llamadas a métodos base

El método base **On Web Authentication** se llama automáticamente cuando una solicitud o procesamiento requiere la ejecución de algún código 4D (excepto para las llamadas REST). También se llama cuando el servidor web recibe una URL estática no válida (por ejemplo, si la página estática solicitada no existe).

Por tanto, se llama al método base **On Web Authentication**:

- cuando el servidor web recibe una URL que solicita un recurso que no existe
- cuando el servidor web recibe una URL que empieza por **4DACTION/**, **4DCGI/** ...
- cuando el servidor web recibe una URL de acceso a la raíz y no se ha definido ninguna página de inicio en la Configuración o mediante el comando **WEB SET HOME PAGE**
- cuando el servidor web procesa una etiqueta que ejecuta código (por ejemplo, **4DSCRIPT**) en una página semidinámica.

Por tanto, NO se llama al método base **On Web Authentication**:

- cuando el servidor web recibe una URL que solicita una página estática válida.
- cuando el servidor web recibe una URL que empieza por **rest/** y se lanza el servidor REST (en este caso, la autenticación se gestiona a través del método base **On REST Authentication** o las **Propiedades de la base**).

### Sintaxis

**On Web Authentication( \$1 : Text ; \$2 : Text ; \$3 : Text ; \$4 : Text ; \$5 : Text ; \$6 : Text ) -> \$0 : Boolean**

Parámetros	Tipo		Descripción
\$1	Texto	<-	URL
\$2	Texto	<-	Encabezados HTTP + cuerpo HTTP (hasta un límite de 32 kb)
\$3	Texto	<-	Dirección IP del cliente web (navegador)
\$4	Texto	<-	Dirección IP del servidor
\$5	Texto	<-	Nombre de usuario
\$6	Texto	<-	Contraseña
\$0	Booleano	->	True = solicitud aceptada, False = solicitud rechazada

Debe declarar estos parámetros de la siguiente manera:

```
//Método base On Web Authentication
```

```
C_TEXT($1;$2;$3;$4;$5;$6)  
C_BOOLEAN($0)
```

```
//Código para el método
```

Como alternativa, puede utilizar la sintaxis [parámetros nombrados](#):

```
// Método base On Web Authentication
#DECLARE ($url : Text; $header : Text; \
$BrowserIP : Text; $ServerIP : Text; \
$user : Text; $password : Text) \
-> $RequestAccepted : Boolean
```

Todos los parámetros del método base `On Web Authentication` no están necesariamente rellenos. La información recibida por el método base depende del [modo de autenticación](#) seleccionado).

## \$1 - URL

El primer parámetro (`$1`) es la URL recibida por el servidor, de la que se ha eliminado la dirección del host.

Tomemos el ejemplo de una conexión a la Intranet. Supongamos que la dirección IP de su máquina 4D Web Server es 123.45.67.89. La siguiente tabla muestra los valores de `$1` en función de la URL introducida en el navegador web:

URL introducida en el navegador web	Valor del parámetro <code>\$1</code>
123.45.67.89	/
<a href="http://123.45.67.89">http://123.45.67.89</a>	/
123.45.67.89/Customers	/Customers
<a href="http://123.45.67.89/Customers/Add">http://123.45.67.89/Customers/Add</a>	/Customers/Add
123.45.67.89/Do_This/If_OK/Do_That	/Do_This/If_OK/Do_That

## \$2 - Encabezado y cuerpo de la petición HTTP

El segundo parámetro (`$2`) es el encabezado y el cuerpo de la petición HTTP enviada por el navegador web. Tenga en cuenta que esta información se pasa a su método base `On Web Authentication` tal cual. Su contenido variará en función de la naturaleza del navegador web que intenta la conexión.

Si su aplicación utiliza esta información, deberá analizar el encabezado y el cuerpo. Puede utilizar los comandos `WEB GET HTTP HEADER` y `WEB GET HTTP BODY`.

Por razones de rendimiento, el tamaño de los datos que pasan por el parámetro `$2` no debe superar los 32 KB. Más allá de este tamaño, son truncados por el servidor HTTP de 4D.

## \$3 - Dirección IP del cliente web

El parámetro `$3` recibe la dirección IP de la máquina del navegador. Esta información puede permitirle distinguir entre las conexiones a la intranet y a Internet.

4D devuelve las direcciones IPv4 en un formato híbrido IPv6/IPv4 escrito con un prefijo de 96 bits, por ejemplo `::ffff:192.168.2.34` para la dirección IPv4 192.168.2.34. Para más información, consulte la sección [Soporte IPv6](#).

## \$4 - Dirección IP del servidor

El parámetro `$4` recibe la dirección IP utilizada para llamar al servidor web. 4D permite el multi-homing, que permite explotar máquinas con más de una dirección IP. Para más información, consulte la [página Configuración](#).

## \$5 y \$6 - Nombre de usuario y contraseña

Los parámetros `$5` y `$6` reciben el nombre de usuario y la contraseña introducidos por el usuario en la caja de diálogo de identificación estándar que muestra el navegador. Esta caja de diálogo aparece para cada conexión, si se

selecciona la autenticación [basic](#) o [digest](#).

Si el nombre de usuario enviado por el navegador existe en 4D, el parámetro \$6 (la contraseña del usuario) no se devuelve por razones de seguridad.

## Parámetro \$0

El método base `On Web Authentication` devuelve un booleano en \$0:

- Si \$0 es True, la conexión es aceptada.
- Si \$0 es False, la conexión es rechazada.

El método base `On Web Connection` sólo se ejecuta si la conexión ha sido aceptada por `On Web Authentication`.

### ADVERTENCIA

Si no se define ningún valor en \$0 o si \$0 no está definido en el método base `On Web Authentication`, la conexión se considera aceptada y se ejecuta el método base `On Web Connection`.

- No llame a ningún elemento de la interfaz en el método base `On Web Authentication` (`ALERT`, `DIALOG`, etc.) porque, de lo contrario, se interrumpirá su ejecución y se rechazará la conexión. Lo mismo ocurrirá si se produce un error durante su procesamiento.

## Ejemplo

Ejemplo del método base `On Web Authentication` en [Modo DIGEST](#):

```
// Método bas On Web Authentication
#DECLARE ($url : Text; $header : Text; $ipB : Text; $ipS : Text; \
    $user : Text; $pw : Text) -> $valid : Boolean

var $found : cs.WebUserSelection
$valid:=False

$found:=ds.WebUser.query("User === :1";$user)
If($found.length=1) // User is found
    $valid:=WEB Validate digest($user;[WebUser]password)
Else
    $valid:=False // El usuario no existe
End if
```

# Sesiones usuario

El servidor web de 4D ofrece funciones integradas para la gestión de sesiones de usuario. La creación y el mantenimiento de sesiones de usuario le permiten controlar y mejorar la experiencia del usuario en su aplicación web. Cuando se activan las sesiones de usuario, los clientes web pueden reutilizar el mismo contexto de servidor de una solicitud a otra.

Las sesiones de usuario del servidor web permiten:

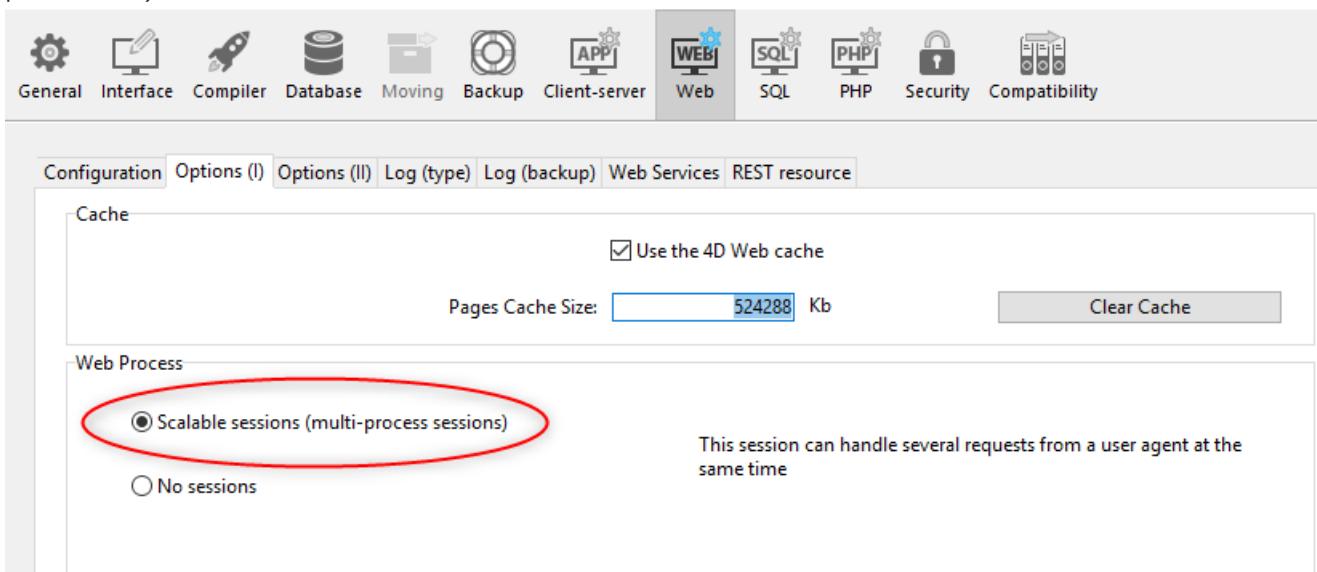
- manejar múltiples peticiones simultáneamente desde el mismo cliente web a través de un número ilimitado de procesos apropiativos (las sesiones del servidor web son escalables),
- compartir datos entre los procesos de un cliente web,
- asociar privilegios a las sesiones de usuario,
- gestionan el acceso a través de un objeto `Session` y de la [Session API](#).

Nota: la implementación actual es sólo el primer paso de una próxima funcionalidad completa que permitirá a los desarrolladores gestionar los permisos jerárquicos de los usuarios a través de las sesiones en toda la aplicación web.

## Activación de sesiones

La funcionalidad de gestión de sesiones puede ser activada y desactivada en su servidor web 4D. Hay diferentes maneras de habilitar la gestión de la sesión:

- Utilizando la opción Sesiones escalables en la página "Web/Opciones (I)" de la Configuración (configuración permanente):



Esta opción está seleccionada por defecto en los nuevos proyectos. Sin embargo, se puede desactivar seleccionando la opción Sin sesiones, en cuyo caso las funcionalidades de la sesión web se desactivan (no hay ningún objeto `Session` disponible).

- Utilizando la propiedad `.scalableSession` del objeto Web Server (para pasar el parámetro `settings` de la función `.start()`). En este caso, esta configuración anula la opción definida en la caja de diálogo Configuración del objeto Servidor Web (no se almacena en el disco).

El comando `WEB SET OPTION` también puede establecer el modo de sesión para el servidor web principal.

En cualquier caso, la configuración es local para la máquina; por lo que puede ser diferente en el servidor web de 4D Server y en los servidores web de las máquinas 4D remotas.

Compatibilidad: una opción Sesiones legacy está disponible en proyectos creados con una versión de 4D anterior a 4D v18 R6 (para más información, consulte el sitio web [doc.4d.com](http://doc.4d.com)).

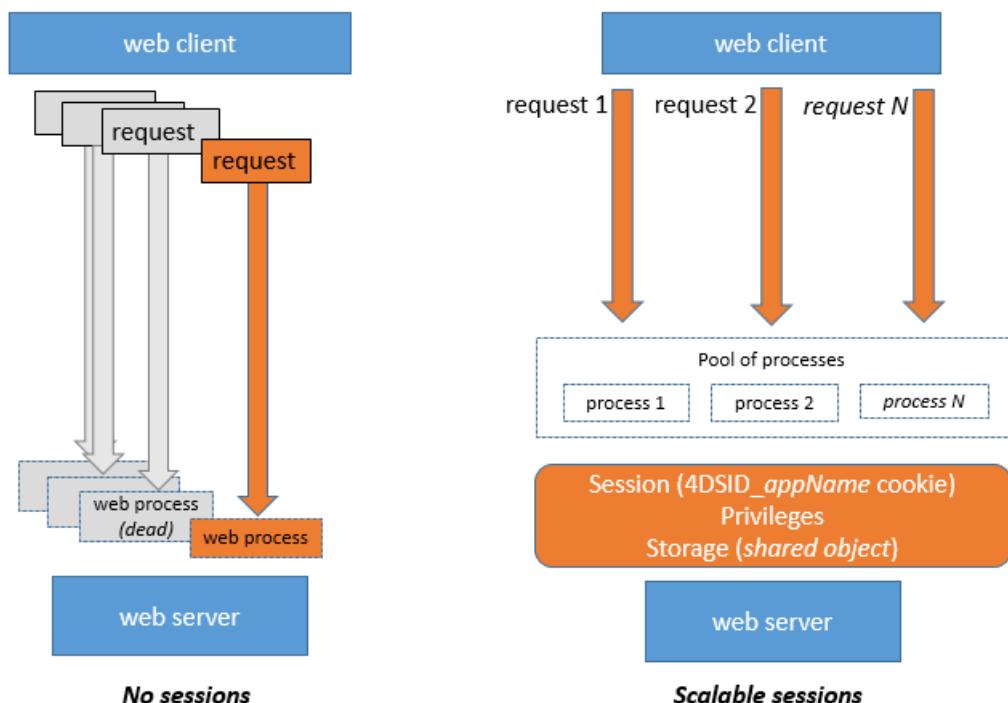
## Implementación de la sesión

Cuando [se habilitan las sesiones](#), se implementan mecanismos automáticos, basados en una cookie privada establecida por el propio 4D: "4DSID\_AppName", donde *AppName* es el nombre del proyecto de la aplicación. Esta cookie hace referencia a la sesión web actual de la aplicación.

El nombre de la cookie puede obtenerse utilizando la propiedad `.sessionCookieName`.

1. En cada petición del cliente web, el servidor web comprueba la presencia y el valor de la cookie privada "4DSID\_AppName".
2. Si la cookie tiene un valor, 4D busca la sesión que creó esta cookie entre las sesiones existentes; si se encuentra esta sesión, se reutiliza para la llamada.
3. Si la solicitud del cliente no corresponde a una sesión ya abierta:
  - se crea una nueva sesión con una cookie privada "4DSID\_AppName" en el servidor web
  - se crea un nuevo objeto Guest `Session` dedicado a la sesión web escalable.

El objeto `Session` actual puede entonces ser accedido a través del comando `Session` en el código de cualquier proceso web.



Los procesos web no suelen terminar, sino que se reciclan en un fondo común para ser más eficientes. Cuando un proceso termina de ejecutar una petición, se devuelve al pool y queda disponible para la siguiente petición. Dado que un proceso web puede ser reutilizado por cualquier sesión, [las variables de proceso](#) deben ser borradas por su código al final de su ejecución (utilizando `CLEAR VARIABLE` por ejemplo). Esta limpieza es necesaria para cualquier información relacionada con el proceso, como una referencia a un archivo abierto. Esta es la razón por la que se recomienda utilizar el objeto `Sesión` cuando se quiera guardar información relacionada con la sesión.

### Modo apropiativo

En 4D Server, las sesiones del servidor web se manejan automáticamente a través de procesos apropiativos, incluso en modo interpretado. Debe asegurarse de que el código de su servidor web es [compatible con una ejecución apropiativa](#).

Para depurar el código web interpretado en la máquina del servidor, asegúrese de que el depurador está

adjuntado al servidor o a una máquina remota. Los procesos web pasan entonces al modo cooperativo y se puede depurar el código del servidor web.

Con 4D monopuesto, el código interpretado siempre se ejecuta en modo cooperativo.

## Compartir información

Cada objeto `Session` ofrece una propiedad `.storage` que es un [objeto compartido](#). Esta propiedad permite compartir información entre todos los procesos manejados por la sesión.

## Fecha de caducidad de la sesión

Una sesión web escalable se cierra cuando:

- el servidor web está detenido,
- se ha alcanzado el tiempo de espera de la cookie de sesión.

La vida útil de una cookie inactiva es de 60 minutos por defecto, lo que significa que el servidor web cerrará automáticamente las sesiones inactivas después de 60 minutos.

Este tiempo de espera se puede establecer utilizando la propiedad `.idleTimeout` del objeto `Session` (el tiempo de espera no puede ser inferior a 60 minutos).

Cuando se cierra una sesión web escalable, si después se llama al comando `Session`:

- el objeto `Session` no contiene privilegios (es una sesión de invitado)
- la propiedad `.storage` está vacía
- se asocia una nueva cookie de sesión a la sesión

## Privilegios

Los privilegios pueden asociarse a las sesiones. En el servidor web, puede proporcionar un acceso o unas funcionalidades específicas en función de los privilegios de la sesión.

Puede asignar privilegios utilizando la función `.setPrivileges()`. En su código, puede comprobar los privilegios de la sesión para permitir o denegar el acceso utilizando la función `.hasPrivilege()`. Por defecto, las nuevas sesiones no tienen ningún privilegio: son sesiones invitados (la función `.isGuest()` devuelve true).

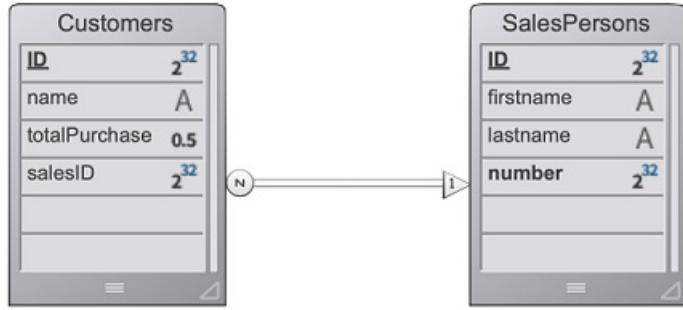
En la implementación actual (v18 R6), sólo está disponible el privilegio "WebAdmin".

Ejemplo:

```
If (Session.hasPrivilege("WebAdmin"))
    //El acceso está concedido, no haga nada
Else
    //Mostrar una página de autenticación
End if
```

## Ejemplo

En una aplicación CRM, cada vendedor gestiona su propia cartera de clientes. El almacén de datos contiene al menos dos clases de datos vinculadas: `Customers` y `SalesPersons` (un vendedor tiene varios clientes).



Queremos que un vendedor se autentique, abra una sesión en el servidor web y que se carguen los 3 primeros clientes en la sesión.

- Ejecutamos esta URL para abrir una sesión:

```
http://localhost:8044/authenticate.shtml
```

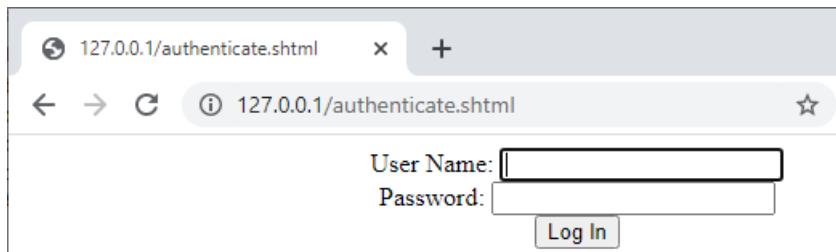
En un entorno de producción, es necesario utilizar una conexión [HTTPS](#) para evitar que cualquier información no cifrada circule por la red.

- La página `authenticate.shtml` es un formulario que contiene los campos de entrada `userId` y `password` y envía una acción `4DACTION POST`:

```

<!DOCTYPE html>
<html>
<body bgcolor="#ffffff">
<FORM ACTION="/4DACTION/authenticate" METHOD=POST>
    UserId: <INPUT TYPE=TEXT NAME=userId VALUE=""><BR>
    Password: <INPUT TYPE=TEXT NAME=password VALUE=""><BR>
<INPUT TYPE=SUBMIT NAME=OK VALUE="Log In">
</FORM>
</body>
</html>

```



- El método `authenticate` project busca la persona `userID` y valida la contraseña contra el valor hash ya almacenado en la tabla `SalesPersons`:

```

var $indexUserId; $indexPassword; $userId : Integer
var $password : Text
var $userTop3; $sales; $info : Object

ARRAY TEXT($anames; 0)
ARRAY TEXT($avalues; 0)

WEB GET VARIABLES($anames; $avalues)

$indexUserId:=Find in array($anames; "userId")
$userId:=Num($avalues{$indexUserId})

$indexPassword:=Find in array($anames; "password")
$password:=$avalues{$indexPassword}

$sales:=ds.SalesPersons.query("userId = :1"; $userId).first()

If ($sales#Null)
    If (Verify password hash($password; $sales.password))
        $info:=New object()
        $info.userName:=$sales.firstname+" "+$sales.lastname
        Session.setPrivileges($info)
        Use (Session.storage)
            If (Session.storage.myTop3=Null)
                $userTop3:=$sales.customers.orderBy("totalPurchase desc").slice(0; 3)
                Session.storage.myTop3:=$userTop3
            End if
        End use
        WEB SEND HTTP REDIRECT("/authenticationOK.shtml")
    Else
        WEB SEND TEXT("This password is wrong")
    End if
Else
    WEB SEND TEXT("This userId is unknown")
End if

```

# Uso de procesos web apropiativos

El servidor web de 4D le permite aprovechar al máximo los ordenadores multinúcleo utilizando procesos web apropiativos en sus aplicaciones. Puede configurar su código relacionado con la web, incluyendo las etiquetas 4D, los métodos base Web o las funciones de clase REST de ORDA para que se ejecuten simultáneamente en tantos núcleos como sea posible.

Para obtener información detallada sobre los procesos apropiativos en 4D, consulte la sección *Procesos 4D apropiativos* del [manual de lenguaje](#).

## Disponibilidad del modo apropiativo para los procesos web

La siguiente tabla indica si el modo apropiativo se utiliza o está disponible, dependiendo del contexto de ejecución:

4D Server	Interpretado ( <a href="#">asociado al depurador</a> )	Interpretado (no asociado al depurador)	Compilado
Servidor REST	cooperativo	apropiativo	apropiativo
Servidor Web	cooperativo	<i>parámetro web</i>	<i>parámetro web</i>
Servidor Web Services	cooperativo	<i>parámetro web</i>	<i>parámetro web</i>

4D remoto/monopuesto	Interpretado	Compilado
Servidor REST	cooperativo	apropiativo
Servidor Web	cooperativo	<i>parámetro web</i>
Servidor Web Services	cooperativo	<i>parámetro web</i>

- Servidor REST: gestiona las [funciones de clase del modelo de datos ORDA](#)
- Servidor web: maneja las [plantillas web](#), [4DACTION](#) y los métodos base
- Servidor de servicios web: gestiona las peticiones SOAP
- *web setting* significa que el modo apropiativo depende de un valor de configuración:
  - cuando la opción [sesiones escalables](#) está seleccionada, el [modo apropiativo se utiliza automáticamente](#) para los procesos web.
  - de lo contrario, la opción [Utilizar procesos apropiativos](#) se tiene en cuenta.
  - en lo que respecta a los procesos de servicios web (servidor o cliente), se soporta el modo apropiativo a nivel del método. Sólo tiene que seleccionar la propiedad "Puede ejecutarse en procesos apropiativos" para los métodos del servidor SOAP publicados (ver [Publicación de un servicio web con 4D](#)) o los métodos del cliente proxy (ver [Suscripción a un servicio web en 4D](#)) y asegurarse de que el compilador confirme que son hilo seguro.

## Escribir código de servidor web hilo seguro

Todo el código 4D ejecutado por el servidor web debe ser hilo seguro si quiere que sus procesos web se ejecuten en modo apropiativo. Cuando el [modo apropiativo está activo](#), las siguientes partes de la aplicación serán evaluadas automáticamente por el compilador 4D:

- Todos los métodos base relacionados con la web:
  - [On Web Authentication](#)
  - [On Web Connection](#)
  - [On REST Authentication](#)
  - [On Mobile App Authentication](#) y [On Mobile App Action](#)

- El método proyecto `compilador_web` (independientemente de su propiedad real "Modo de ejecución");
- Básicamente cualquier código procesado por el comando `PROCESS 4D TAGS` en el contexto web, por ejemplo a través de páginas `.shtml`
- Todo método proyecto con el atributo "Disponible a través de etiquetas 4D y URLs" (`4DACTION`, etc.)
- Triggers para tablas con el atributo "Exponer como recurso REST"
- [funciones de clase del modelo de datos ORDA](#) llamadas vía REST

Para cada uno de estos métodos y partes de código, el compilador comprobará si se respetan las reglas de seguridad de hilos, y devolverá errores en caso de que haya problemas. Para más información sobre las reglas hilo seguro, consulte el párrafo *Escribir un método hilo seguro* en el capítulo *Procesos* del manual de [Lenguaje 4D](#).

## Código web 4D hilo seguro

La mayoría de los comandos y funciones 4D relacionados con la web, los métodos base y las URL son hilo seguro y pueden utilizarse en modo apropiativo.

### Comandos 4D y métodos base

Todos los comandos 4D relacionados con la web son hilo seguro, *es decir*:

- todos los comandos del tema *Servidor Web*,
- todos los comandos del tema *Cliente HTTP*.

Los métodos base relacionados con la web son hilo seguro y pueden utilizarse en modo apropiativo (ver arriba): `On Web Authentication`, `On Web Connection`, `On REST Authentication` ...).

Por supuesto, el código ejecutado por estos métodos también debe ser hilo seguro.

### URLs del servidor web

Las siguientes URLs 4D Web Server son hilo seguro y pueden ser utilizadas en modo apropiativo:

- `4daction/` (el método proyecto llamado también debe ser hilo seguro)
- `4dcgi/` (los métodos base llamados también deben ser hilo seguro)
- `4dwebtest/`
- `4dblank/`
- `4dstats/`
- `4dhtmlstats/`
- `4dcache-clear/`
- `rest/`
- `4dimgfield/` (generado por `PROCESS 4D TAGS` para la petición web en los campos imagen)
- `4dimg/` (generado por `PROCESS 4D TAGS` para la petición web en las variables imagen)

### Icono de proceso web apropiativo

Tanto el Explorador de ejecución como la ventana de administración de 4D Server muestran un ícono específico para los procesos web apropiativos:

Tipo de proceso	Icono
Método Web (proceso apropiativo)	

# Comencemos

4D le ofrece un servidor REST poderoso, que permite el acceso directo a los datos almacenados en sus aplicaciones 4D.

El servidor REST está incluido en 4D y 4D Server, está automáticamente disponible en sus aplicaciones 4D [una vez configurado](#).

Esta sección pretende ayudar a familiarizarse con la funcionalidad de REST mediante un ejemplo sencillo. Vamos a:

- crear y configurar un proyecto de aplicación 4D básico
- acceder a los datos del proyecto 4D a través de REST utilizando un navegador estándar.

Para simplificar el ejemplo, vamos a utilizar 4D y un navegador que se ejecutan en la misma máquina. Por supuesto, también puede utilizar una arquitectura remota.

## Crear y configurar el proyecto 4D

1. Lance su aplicación 4D o 4D Server y cree un nuevo proyecto. Puede llamarlo, por ejemplo, "Emp4D".
2. En el editor de estructura, cree una tabla [Employees] y añada los siguientes campos a la misma:
  - Lastname (Alpha)
  - Firstname (Alpha)
  - Salary (Longint)

Employees	
ID	2 <sup>32</sup>
Lastname	A
Firstname	A
Salary	2 <sup>32</sup>

La opción "Exponer un recurso REST" está marcada por defecto para la tabla y cada campo; no cambie esta configuración.

3. Cree los formularios y, a continuación, cree algunos empleados:

Emp4D - Employees: 3 of 3				
ID :	Lastname :	Firstname :	Salary :	
1	Brown	Michael	25000	
2	Jones	Maryanne	35000	
3	Smithers	Jack	41000	

4. Abra la página Web > Web Features de la caja de diálogo de las Propiedades y [marque la opción Exponer como servidor REST](#).
5. En el menú Ejecutar, seleccione Iniciar el servidor Web (si es necesario), luego seleccione Probar el servidor Web.

4D muestra la página de inicio por defecto del servidor web de 4D.

## Acceso a los datos 4D con el navegador

Ahora puede leer y editar datos dentro de 4D sólo a través de peticiones REST.

Toda petición de URL 4D REST comienza por `/rest`, que se debe insertar después del área `address:port`. Por ejemplo, para ver lo que hay dentro del almacén de datos de 4D, puede escribir:

```
http://127.0.0.1/rest/$catalog
```

El servidor REST responde:

```
{  
    "__UNIQID": "96A49F7EF2ABDE44BF32059D9ABC65C1",  
    "dataClasses": [  
        {  
            "name": "Employees",  
            "uri": "/rest/$catalog/Employees",  
            "dataURI": "/rest/Employees"  
        }  
    ]  
}
```

Esto significa que el almacén de datos contiene la clase de datos Employees. Puede ver los atributos de la clase de datos escribiendo:

```
/rest/$catalog/Employees
```

Si desea obtener todas las entidades de la clase de datos Employee, escriba:

```
/rest/Employees
```

Respuesta:

```
{
  "__entityModel": "Employees",
  "__GlobalStamp": 0,
  "__COUNT": 3,
  "__FIRST": 0,
  "__ENTITIES": [
    {
      "__KEY": "1",
      "__TIMESTAMP": "2020-01-07T17:07:52.467Z",
      "__STAMP": 2,
      "ID": 1,
      "Lastname": "Brown",
      "Firstname": "Michael",
      "Salary": 25000
    },
    {
      "__KEY": "2",
      "__TIMESTAMP": "2020-01-07T17:08:14.387Z",
      "__STAMP": 2,
      "ID": 2,
      "Lastname": "Jones",
      "Firstname": "Maryanne",
      "Salary": 35000
    },
    {
      "__KEY": "3",
      "__TIMESTAMP": "2020-01-07T17:08:34.844Z",
      "__STAMP": 2,
      "ID": 3,
      "Lastname": "Smithers",
      "Firstname": "Jack",
      "Salary": 41000
    }
  ],
  "__SENT": 3
}
```

Tiene muchas posibilidades para filtrar los datos a recibir. Por ejemplo, para obtener sólo el valor del atributo "Lastname" de la 2<sup>a</sup> entidad, basta con escribir:

```
/rest/Employees(2)/Lastname
```

Respuesta:

```
{
  "__entityModel": "Employees",
  "__KEY": "2",
  "__TIMESTAMP": "2020-01-07T17:08:14.387Z",
  "__STAMP": 2,
  "Lastname": "Jones"
}
```

La [API REST](#) ofrece varios comandos para interactuar con las aplicaciones 4D.

# Configuración del servidor

Utilizando peticiones HTTP estándar, el servidor 4D REST permite a las aplicaciones externas acceder directamente a los datos de su aplicación, es decir, recuperar información sobre las clases de datos de su proyecto, manipular datos, entrar en su aplicación web, y mucho más.

Para comenzar a utilizar las funcionalidades REST, es necesario iniciar y configurar el servidor 4D REST.

- En 4D Server, para abrir una sesión REST es necesario disponer de una licencia cliente 4D gratuita.
- En 4D monopuesto, puede abrir hasta tres sesiones REST para realizar pruebas.
- Debe gestionar la [sesión](#) para su aplicación solicitante.

## Iniciar el servidor REST

Por razones de seguridad, por defecto, 4D no responde a las peticiones REST. Si desea iniciar el servidor REST, debe marcar la opción Exponer como servidor REST en la página Web > Web Features de la configuración de la estructura para que se procesen las peticiones REST.

The screenshot shows the 'Git - Structure Settings' interface. At the top, there is a toolbar with icons for General, Interface, Compiler, Database, Moving, Backup, Client-server, Web (which is highlighted), SQL, PHP, Security, and Compatibility. Below the toolbar, a navigation bar includes Configuration, Options (I), Options (II), Log (type), Log (backup), Web Services, and Web features (which is also highlighted). The main area is titled 'Publishing' and contains a note: 'NOTE: This service is only active on 4D Developer Professional and 4D Server.' There is a checked checkbox labeled 'Expose as REST server'. Below this is a section titled 'Access' with a note: 'NOTE: This setting is only taken into account when the 4D password access system is activated (the Designer has been assigned a password) and the database method "On REST Authentication" does not exist.' A dropdown menu for 'Read/Write' is set to '<Anyone>'.

Los servicios REST utilizan el servidor HTTP 4D, por lo que debe asegurarse de que el servidor web 4D está iniciado.

El mensaje de advertencia "Atención, verifique los privilegios de acceso" aparece cuando se marca esta opción para llamar la atención sobre el hecho de que cuando se activan los servicios REST, por defecto el acceso a los objetos de la base de datos es gratuito mientras no se hayan configurado los accesos REST.

Debe reiniciar la aplicación 4D para que los cambios surtan efecto.

## Configuración del acceso REST

Por defecto, los accesos REST están abiertos a todos los usuarios, lo que obviamente no es recomendable por razones de seguridad, y también para controlar el uso de las licencias de los clientes.

Puede configurar los accesos REST de una de las siguientes maneras:

- asignar un grupo de usuarios en lectura/escritura a los servicios REST en la página "Web > Web Features" de los parámetros de estructura;

- escribiendo un método base `On REST Authentication` para interceptar y manejar cada petición REST inicial.

No se pueden utilizar ambas funciones simultáneamente. Una vez que se ha definido un método base `On REST Authentication`, 4D delega totalmente el control de las peticiones REST en él: se ignora cualquier ajuste realizado mediante el menú "Lectura/Escritura" de la página de Web>Funcionalidades Web de los parámetros de la estructura.

## Uso de los parámetros de la Estructura

El menú Lectura/Escritura de la página "Web > Web Features" de los parámetros de la estructura indica un grupo de usuarios 4D que está autorizado a establecer el enlace con la aplicación 4D utilizando las peticiones REST.

Por defecto, el menú muestra `<Anyone>`, lo que significa que los accesos REST están abiertos a todos los usuarios. Una vez que haya especificado un grupo, sólo una cuenta de usuario de 4D que pertenezca a este grupo podrá ser utilizada para [acceder a 4D mediante una petición REST](#). Si se utiliza una cuenta que no pertenece a este grupo, 4D devuelve un error de autenticación al remitente de la petición.

Para que esta configuración tenga efecto, el método base `On REST Authentication` no debe estar definido. Si existe, 4D ignora los parámetros de acceso definidos en las propiedades de la estructura.

## Método base On REST Authentication

El método base `On REST Authentication` le ofrece una forma personalizada de controlar la apertura de sesiones REST en 4D. Este método base se llama automáticamente cuando se abre una nueva sesión a través de una solicitud REST. Cuando se recibe una [solicitud para abrir una sesión REST](#), los identificadores de conexión se ofrecen en el encabezado de la solicitud. Se llama al método base `On REST Authentication` para poder evaluar estos identificadores. Puede utilizar la lista de usuarios de la aplicación 4D o puede utilizar su propia tabla de identificadores. Para obtener más información, consulte la [documentación](#) del método base `On REST Authentication`.

## Exponer tablas y campos

Una vez activados los servicios REST en la aplicación 4D, por defecto una sesión REST puede acceder a todas las tablas y campos de la base de datos 4D a través de la [interfaz del datastore](#). Así, puede utilizar sus datos. Por ejemplo, si su base de datos contiene una tabla [Employee], es posible escribir:

```
http://127.0.0.1:8044/rest/Employee/?$filter="salary>10000"
```

Esta petición devolverá todos los empleados cuyo campo de salario sea superior a 10000.

Las tablas y/o campos 4D que tienen el atributo "Invisible" también se exponen en REST por defecto.

Si desea personalizar los objetos del datastore accesibles a través de REST, debe desactivar la exposición de cada tabla y/o campo que deseé ocultar. Cuando una petición REST intenta acceder a un recurso no autorizado, 4D devuelve un error.

## Exponer las tablas

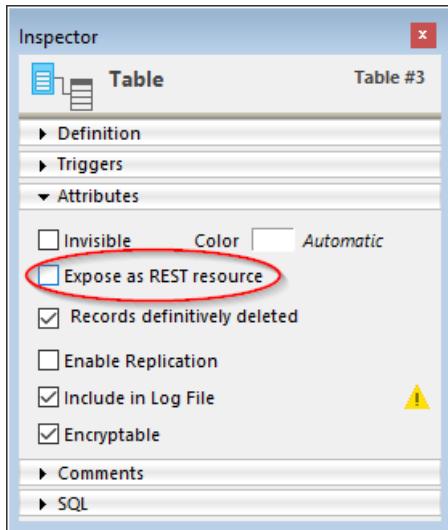
Por defecto, todas las tablas se exponen en REST.

Por razones de seguridad, es posible que desee exponer sólo ciertas tablas de su almacén de datos a las llamadas REST. Por ejemplo, si ha creado una tabla [Users] que almacena los nombres de usuario y las contraseñas, sería mejor no exponerla.

Para eliminar la exposición REST de una tabla:

1. Visualice el inspector de tablas en el editor de estructuras y seleccione la tabla que desea modificar.

2. Desmarque la opción Exponer como recurso REST:



Haga esto para cada tabla cuya exposición deba ser modificada.

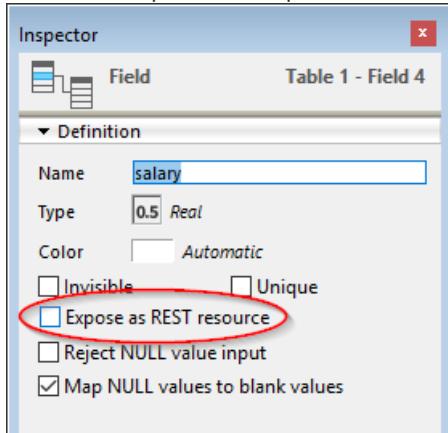
## Exponer los campos

Por defecto, todos los campos de una base 4D se exponen en REST.

Es posible que no quiera exponer ciertos campos de sus tablas a REST. Por ejemplo, es posible que no quiera exponer el campo [Employees]Salary.

Para eliminar la exposición REST de un campo:

1. Visualice el inspector de campo en el editor de estructuras y seleccione el campo a modificar.
2. Desmarque la opción Exponer como recurso REST para el campo.



Repita esta operación para cada campo cuya exposición deba modificarse.

Para que un campo sea accesible a través de REST, la tabla padre también debe serlo. Si la tabla padre no está expuesta, ninguno de sus campos lo estará, independientemente de su estado.

# Usuarios y sesiones

REST requests can benefit from [web user sessions](#), providing extra features such as multiple requests handling, data sharing between the web client processes, and user privileges.

Como primer paso para abrir una sesión REST en el servidor 4D, el usuario que envía la solicitud debe estar autenticado.

## Autenticación de los usuarios

You log in a user to your application by calling `$directory/login` in a POST request including the user's name and password in the header. This request calls the `On REST Authentication` database method (if it exists), where you can check the user's credentials (see example below).

## Apertura de sesiones

When [scalable sessions are enabled](#) (recommended), if the `On REST Authentication` database method returns `true`, a user session is then automatically opened and you can handle it through the `Session` object and the [Session API](#). Subsequent REST requests will reuse the same session cookie.

If the `On REST Authentication` database method has not been defined, a `guest` session is opened.

## Ejemplo

In this example, the user enters their email and password in an html page that requests `$directory/login` in a POST (it is recommended to use an HTTPS connection to send the html page). The `On REST Authentication` database method is called to validate the credentials and to set the session.

Página de inicio de sesión HTML:

The image shows a simple HTML form for user authentication. It consists of three elements: a text input field labeled "Email:", a text input field labeled "Password:", and a red "Login" button. All elements are contained within a single vertical column.

Email:	<input type="text"/>
Password:	<input type="password"/>
<input type="button" value="Login"/>	

```

<html><body bgcolor="#ffffff">

<div id="demo">
  <FORM name="myForm">
    Email: <INPUT TYPE=TEXT NAME=userId VALUE=""><BR>
    Password: <INPUT TYPE=TEXT NAME=password VALUE=""><BR>
    <button type="button" onclick="onClick()">
      Login
    </button>
  </div id="authenticationFailed" style="visibility:hidden;">Authentication failed</div>
</FORM>
</div>

<script>
function sendData(data) {
  var XHR = new XMLHttpRequest();

  XHR.onreadystatechange = function() {
    if (this.status == 200) {
      window.location = "authenticationOK.shtml";
    }
    else {
      document.getElementById("authenticationFailed").style.visibility = "visible";
    }
  };
}

XHR.open('POST', 'http://127.0.0.1:8044/rest/$directory/login'); //rest server address

XHR.setRequestHeader('username-4D', data.userId);
XHR.setRequestHeader('password-4D', data.password);
XHR.setRequestHeader('session-4D-length', data.timeout);

XHR.send();
};

function onClick()
{
sendData({userId:document.forms['myForm'].elements['userId'].value , password:document.forms['myForm'].e
}
</script></body></html>

```

When the login page is sent to the server, the `On REST Authentication` database method is called:

```

//On REST Authentication

#DECLARE($userId : Text; $password : Text) -> $Accepted : Boolean
var $sales : cs.SalesPersonsEntity

$Accepted:=False

//A '/rest' URL has been called with headers username-4D and password-4D
If ($userId#"")
  $sales:=ds.SalesPersons.query("email = :1"; $userId).first()
  If ($sales#Null)
    If (Verify password hash($password; $sales.password))
      fillSession($sales)
      $Accepted:=True
    End if
  End if
End if

```

As soon as it has been called and returned `True`, the `On REST Authentication` database method is no longer called in the session.

The `fillSession` project method initializes the user session, for example:

```
#DECLARE($sales : cs.SalesPersonsEntity)
var $info : Object

$info:=New object()
$info.userName:=$sales.firstname" "+$sales.lastname

Session.setPrivileges($info)

Use (Session.storage)
If (Session.storage.myTop3=NULL)
    Session.storage.myTop3:=$sales.customers.orderBy("totalPurchase desc").slice(0; 3)
End if
End use
```

# Obtener información del servidor

Puede obtener varias informaciones del servidor REST:

- los almacenes de datos expuestos y sus atributos
- el contenido de la caché del servidor REST, incluidas las sesiones de los usuarios.

## Catálogo

Utilice los parámetros `$catalog`, `$catalog/{dataClass}`, o `$catalog/$all` para obtener la lista de [las clases de datos expuestas y sus atributos](#).

Para obtener la colección de todas las clases de datos expuestas junto con sus atributos:

```
GET /rest/$catalog/$all
```

## Información de la caché

Utilice el parámetro `$info` para obtener información sobre las selecciones de entidades actualmente almacenadas en la caché de 4D Server, así como sobre las sesiones de usuario en ejecución.

## queryPath y queryPlan

Las selecciones de entidades generadas a través de búsquedas pueden tener las dos propiedades siguientes: `queryPlan` y `queryPath`. Para calcular y devolver estas propiedades, basta con añadir `$queryPlan` y/o `$queryPath` en la petición REST.

Por ejemplo:

```
GET /rest/People/$filter="employer.name=acme AND lastName=Jones"&$queryplan=true&$querypath=true
```

Estas propiedades son objetos que contienen información sobre cómo el servidor realiza consultas compuestas internamente a través de clases de datos y relaciones:

- `queryPlan`: objeto que contiene la descripción detallada de la petición justo antes de ser ejecutada (es decir, la consulta planificada).
- `queryPath`: objeto que contiene la descripción detallada de la consulta tal y como se ha realizado realmente.

La información registrada incluye el tipo de consulta (indexada y secuencial) y cada subconsulta necesaria junto con los operadores de conjunción. Las rutas de acceso de las peticiones también contienen el número de entidades encontradas y el tiempo necesario para ejecutar cada criterio de búsqueda. Puede resultarle útil analizar esta información mientras desarrolla su aplicación. Generalmente, la descripción del plan de consulta y su ruta de acceso son idénticas, pero pueden diferir porque 4D puede implementar optimizaciones dinámicas cuando se ejecuta una consulta para mejorar el rendimiento. Por ejemplo, el motor 4D puede convertir dinámicamente una consulta indexada en una secuencial si estima que es más rápida. Este caso concreto puede darse cuando el número de entidades que se buscan es bajo.

# Manipulación de datos

Todos los atributos, dataclasses expuestos y todas las funciones pueden ser accedidos a través de REST. Los nombres de clases de datos, de atributos y de funciones son sensibles a las mayúsculas y minúsculas; sin embargo, los datos de las búsquedas no lo son.

## Buscar datos

Para consultar los datos directamente, puede hacerlo mediante la función `$filter`. Por ejemplo, para encontrar a una persona llamada "Smith", podría escribir:

```
http://127.0.0.1:8081/rest/Person/?$filter="lastName=Smith"
```

## Añadir, modificar y eliminar entidades

Con la API REST, puede realizar todas las manipulaciones a los datos como puede hacerlo en 4D.

Para añadir y modificar entidades, puede llamar a `$method=update`. Si desea eliminar una o varias entidades, puede utilizar `$method=delete`.

Además la recuperación de una sola entidad en una clase de datos utilizando `{dataClass}({key})`, también puede escribir una función de clase que devuelva una entity selection (o una colección).

Antes de devolver una selección, también puede ordenarla utilizando `$orderby` uno o varios atributos (incluso los atributos de relación).

## Navegación de datos

Añada el `$skip` (para definir con qué entidad empezar) y `$top/$limit` (para definir cuántas entidades devolver) de las peticiones REST a sus consultas o selecciones de entidades para navegar por la colección de entidades.

## Creación y gestión del conjunto de entidades

Un conjunto de entidades (también conocido como *selección de entidades*) es una colección de entidades obtenidas a través de una petición REST que se almacena en la caché de 4D Server. El uso de un conjunto de entidades evita que se consulte continuamente la aplicación para obtener los mismos resultados. El acceso a un conjunto de entidades es mucho más rápido y puede mejorar la velocidad de su aplicación.

Para crear un conjunto de entidades, llame a `$method=entityset` en su solicitud REST. Como medida de seguridad, también se puede utilizar `$savedfilter` y/o `$savedorderby` cuando se llame a `$filter` y/o `$orderby` para que si alguna vez el conjunto de entidades se agota o se elimina del servidor, se pueda recuperar rápidamente con el mismo ID que antes.

Para acceder al conjunto de entidades, debe utilizar `$entityset/{entitySetID}`, por ejemplo:

```
/rest/People/$entityset/0AF4679A5C394746BFEB68D2162A19FF
```

Por defecto, un conjunto de entidades se almacena durante dos horas; sin embargo, puede cambiar el tiempo de espera pasando un nuevo valor a `$timeout`. El tiempo de espera se restablece continuamente al valor definido para su tiempo de espera (ya sea el predeterminado o el que usted defina) cada vez que lo utilice.

Si desea eliminar un conjunto de entidades de la caché de 4D Server, puede utilizar `$method=release`.

Si se modifica alguno de los atributos de la entidad en el conjunto de entidades, los valores se actualizarán. Sin embargo, si se modifica un valor que formaba parte de la consulta ejecutada para crear el conjunto de entidades, no se eliminará del conjunto de entidades aunque ya no se ajuste a los criterios de búsqueda. Las entidades que elimine, por supuesto, dejarán de formar parte del conjunto de entidades.

Si el conjunto de entidades ya no existe en la caché de 4D Server, se recreará con un nuevo tiempo de espera por defecto de 10 minutos. El conjunto de entidades se refrescará (pueden incluirse ciertas entidades y eliminarse otras) desde la última vez que se creó, si ya no existía antes de recrearlo.

Utilizando `$entityset/{entitySetID}?$logicOperator... &$otherCollection`, puede combinar dos conjuntos de entidades que haya creado previamente. Puede combinar los resultados en ambos, devolver sólo lo que es común entre los dos, o devolver lo que no es común entre los dos.

Se devuelve una nueva selección de entidades; sin embargo, también se puede crear un nuevo conjunto de entidades llamando `$method=entityset` al final de la petición REST.

## Cálculo de datos

Utilizando `$compute`, se puede calcular el promedio, el count, el min, el max, o la suma para un atributo específico en una clase de datos. También puede calcular todos los valores con la palabra clave `$all`.

Por ejemplo, para obtener el salario más alto:

```
/rest/Employee/salary/?$compute=max
```

Para calcular todos los valores y devolver un objeto JSON:

```
/rest/Employee/salary/?$compute=$all
```

## Llamar las funciones de clase del modelo de datos

Puede llamar las [funciones de clase usuarios](#) ORDA del modelo de datos vía las peticiones POST, para poder beneficiarse del API de la aplicación objetivo. Por ejemplo, si ha definido una función `getCity()` en la dataclass City, podría llamarla utilizando la siguiente petición:

```
/rest/City/getCity
```

con los datos en el cuerpo de la petición: `["Paris"]`

Las llamadas a los métodos proyecto 4D que se exponen como servicio REST aún se soportan, pero son obsoletas.

## Selección de atributos a obtener

Siempre se puede definir qué atributos devolver en la respuesta REST después de una solicitud inicial pasando su ruta en la solicitud (*por ejemplo*, `Company(1)/name,revenues/`)

Puede aplicar este filtro de las siguientes maneras:

Objeto	Sintaxis	Ejemplo
Dataclass	<code>{dataClass}/{att1,att2...}</code>	<code>/People/firstName,lastName</code>
Collection de entidades	<code>{dataClass}/{att1,att2...}/?\$filter="{filter}"</code>	<code>/People/firstName,lastName/?\$filter="lastName=</code>
Entidad específica	<code>{dataClass}({ID})/{att1,att2...}</code>	<code>/People(1)/firstName,lastName</code>
	<code>{dataClass}:{attribute}(value)/{att1,att2...}/</code>	<code>/People:firstName(Larry)/firstName,lastName/</code>
Entity selection	<code>{dataClass}/{att1,att2...}/\$entityset/{entitySetID}</code>	<code>/People/firstName/\$entityset/528BF90F1089491!</code>

Los atributos deben estar delimitados por una coma, *i.e.*, `/Employee/firstName,lastName,salary`. Se pueden pasar atributos de almacenamiento o relacionales.

## Ejemplos

A continuación se presentan algunos ejemplos, que muestran cómo especificar qué atributos devolver en función de la técnica utilizada para recuperar las entidades.

Puede aplicar esta técnica a:

- Clases de datos (todas o una colección de entidades en una clase de datos)
- Entidades específicas
- Conjunto de entidades

### Ejemplo con una dataclass

Las siguientes peticiones devuelven sólo el nombre y el apellido de la clase de datos People (ya sea toda la clase de datos o una selección de entidades basada en la búsqueda definida en `$filter` ).

```
GET /rest/People/firstName,lastName/
```

Resultado:

```
{
  __entityModel: "People",
  __COUNT: 4,
  __SENT: 4,
  __FIRST: 0,
  __ENTITIES: [
    {
      __KEY: "1",
      __STAMP: 1,
      firstName: "John",
      lastName: "Smith"
    },
    {
      __KEY: "2",
      __STAMP: 2,
      firstName: "Susan",
      lastName: "O'Leary"
    },
    {
      __KEY: "3",
      __STAMP: 2,
      firstName: "Pete",
      lastName: "Marley"
    },
    {
      __KEY: "4",
      __STAMP: 1,
      firstName: "Beth",
      lastName: "Adams"
    }
  ]
}
```

```
GET /rest/People/firstName,lastName/?$filter="lastName='A@'"/
```

Resultado:

```
{
  __entityModel: "People",
  __COUNT: 1,
  __SENT: 1,
  __FIRST: 0,
  __ENTITIES: [
    {
      __KEY: "4",
      __STAMP: 4,
      firstName: "Beth",
      lastName: "Adams"
    }
  ]
}
```

## Ejemplo de entidad

La siguiente petición devuelve sólo los atributos de nombre y apellido de una entidad específica de la clase de datos People:

```
GET /rest/People(3)/firstName,lastName/
```

Resultado:

```
{
  __entityModel: "People",
  __KEY: "3",
  __STAMP: 2,
  firstName: "Pete",
  lastName: "Marley"
}
```

```
GET /rest/People(3)/
```

Resultado:

```
{
  __entityModel: "People",
  __KEY: "3",
  __STAMP: 2,
  ID: 3,
  firstName: "Pete",
  lastName: "Marley",
  salary: 30000,
  employer: {
    __deferred: {
      uri: "http://127.0.0.1:8081/rest/Company(3)",
      __KEY: "3"
    }
  },
  fullName: "Pete Marley",
  employerName: "microsoft"
}
```

## Ejemplo de conjunto de entidades

Una vez que haya [creado un conjunto de entidades](#), puede filtrar la información que contiene definiendo qué atributos debe devolver:

```
GET /rest/People/firstName,employer.name/$entityset/BDCD8AABE13144118A4CF8641D5883F5?$expand=employer
```

## Visualización de un atributo de imagen

Si desea visualizar un atributo de imagen en su totalidad, escriba lo siguiente:

```
GET /rest/Employee(1)/photo?$imageformat=best&$version=1&$expand=photo
```

Para más información sobre los formatos de imagen, consulte [\\$imageformat](#). Para más información sobre el parámetro de versión, consulte [\\$version](#).

## Guardar un atributo BLOB en el disco

Si quiere guardar un BLOB almacenado en su clase de datos, puedes escribir lo siguiente:

```
GET /rest/Company(11)/blobAtt?$binary=true&$expand=blobAtt
```

## Recuperar sólo una entidad

Puede utilizar la sintaxis [{dataClass}:{attribute}\(value\)](#) cuando quiera recuperar sólo una entidad. Es especialmente útil cuando se quiere hacer una búsqueda relacionada que no se crea en la llave primaria de la clase de datos. Por ejemplo, puede escribir:

```
GET /rest/Company:companyCode("Acme001")
```

# Llamar a las funciones de clase ORDA

Puede llamar a [funciones de clase de modelos de datos](#) definidas para el modelo de datos ORDA a través de sus peticiones REST, para poder beneficiarse de la API expuesta de la aplicación 4D objetivo.

Las funciones se llaman simplemente en peticiones POST en la interfaz ORDA apropiada, sin (). Por ejemplo, si ha definido una función `getCity()` en la dataclass City, podría llamarla utilizando la siguiente petición:

```
/rest/City/getCity
```

con los datos en el cuerpo de la petición POST: `["Aguada"]`

En el lenguaje 4D, esta llamada equivale a:

```
$city:=ds.City.getCity("Aguada")
```

Sólo las funciones con la palabra clave `exposed` pueden ser llamadas directamente desde las peticiones REST. Ver la sección [Funciones expuestas vs. no expuestas](#).

## Llamadas de las funciones

Las funciones deben llamarse siempre utilizando peticiones POST (una petición GET recibirá un error).

Las funciones son llamadas en el objeto correspondiente en el almacén de datos del servidor.

Función de clase	Sintaxis
<code>datastore class</code>	<code>/rest/\$catalog/DataStoreClassFunction</code>
<code>dataclass class</code>	<code>/rest/{dataClass}/DataClassClassFunction</code>
<code>entitySelection class</code>	<code>/rest/{dataClass}/EntitySelectionClassFunction</code>
	<code>/rest/{dataClass}/EntitySelectionClassFunction/\$entityset/entitySetNumber</code>
	<code>/rest/{dataClass}/EntitySelectionClassFunction/\$filter</code>
	<code>/rest/{dataClass}/EntitySelectionClassFunction/\$orderby</code>
<code>entity class</code>	<code>/rest/{dataClass}(key)/EntityClassFunction/</code>

`/rest/{dataClass}/Function` puede utilizarse para llamar a una función de dataclass o de selección de entidades (`/rest/{dataClass}` devuelve todas las entidades de la DataClass como una selección de entidades). La función se busca primero en la clase de selección de entidades. Si no se encuentra, se busca en la dataclass. En otras palabras, si una función con el mismo nombre se define tanto en la clase DataClass como en la clase EntitySelection, la función de clase de DataClass nunca se ejecutará.

Todo el código 4D llamado desde las peticiones REST debe ser hilo seguro si el proyecto se ejecuta en modo compilado, porque el Servidor REST siempre utiliza procesos apropiativos en este caso (el valor de la propiedad [Utilizar proceso apropiativo](#) es ignorado por el Servidor REST).

## Parámetros

Puede enviar los parámetros a las funciones definidas en las clases usuarios ORDA. Del lado del servidor, se recibirán en las funciones de clase en los parámetros normales \$1, \$2, etc.

Se aplican las siguientes reglas:

- Los parámetros deben pasarse en el cuerpo de la petición POST
- Los parámetros deben estar incluidos en una colección (formato JSON)
- Todos los tipos de datos escalares soportados en las colecciones JSON pueden ser pasados como parámetros.
- La selección de entidades y la entidad se pueden pasar como parámetros. El objeto JSON debe contener atributos específicos utilizados por el servidor REST para asignar datos a los objetos ORDA correspondientes: \_\_DATACLASS, \_\_ENTITY, \_\_ENTITIES, \_\_DATASET.

Ver [este ejemplo](#) y [este ejemplo](#).

## Parámetro de valor escalar

El(s) parámetros deben estar simplemente incluirse en una colección definida en el cuerpo. Por ejemplo, con una función de dataclass `getCities()` que recibe parámetros de tipo texto: `/rest/City/getCities`

Parámetros en el cuerpo: `["Aguada","Paris"]`

Todos los tipos de datos JSON son soportados en los parámetros, incluidos los punteros JSON. Las fechas se pueden pasar como cadenas en formato de fecha ISO 8601 (por ejemplo, "2020-08-22T22:00:00Z").

## Parámetro de entidad

Las entidades pasadas en los parámetros son referenciadas en el servidor a través de su llave (*es decir*; propiedad `__KEY`). Si el parámetro llave se omite en una petición, una nueva entidad se carga en memoria del servidor. También puede pasar valores para todos los atributos de la entidad. Estos valores se utilizarán automáticamente para la entidad manejada en el servidor.

Si la petición envía los valores de atributo modificados para una entidad existente en el servidor, la función de modelo de datos ORDA llamada se ejecutará automáticamente en el servidor con los valores modificados. Esta funcionalidad le permite, por ejemplo, verificar el resultado de una operación en una entidad, tras aplicar todas las reglas de negocio, desde la aplicación cliente. A continuación, puede decidir guardar o no la entidad en el servidor.

Propiedades	Tipo	Descripción
Atributos de la entidad	mixto	Opcional - Valores a modificar
<code>__DATACLASS</code>	Cadena	Obligatorio - Indica la Dataclass de la entidad
<code>__ENTITY</code>	Booleano	Obligatorio - True para indicar al servidor que el parámetro es una entidad
<code>__KEY</code>	mixto (mismo tipo que la llave primaria)	Opcional - Llave primaria de la entidad

- Si no se proporciona `__KEY`, se crea una nueva entidad en el servidor con los atributos dados.
- Si se proporciona `__KEY`, la entidad correspondiente a `__KEY` se carga en el servidor con los atributos dados

Ver los ejemplos de [creación](#) o de [actualización](#) de las entidades.

## Parámetro de entidad asociado

Las mismas propiedades que para un [parámetro de entidad](#). Además, la entidad relacionada debe existir y ser referenciada por `__KEY`, que contiene su llave primaria.

Ver los ejemplos para [creación](#) o [actualización](#) de las entidades con las entidades relacionadas.

## Parámetro de selección de entidad

La selección de entidades debe haber sido definida previamente utilizando `$method=entityset`.

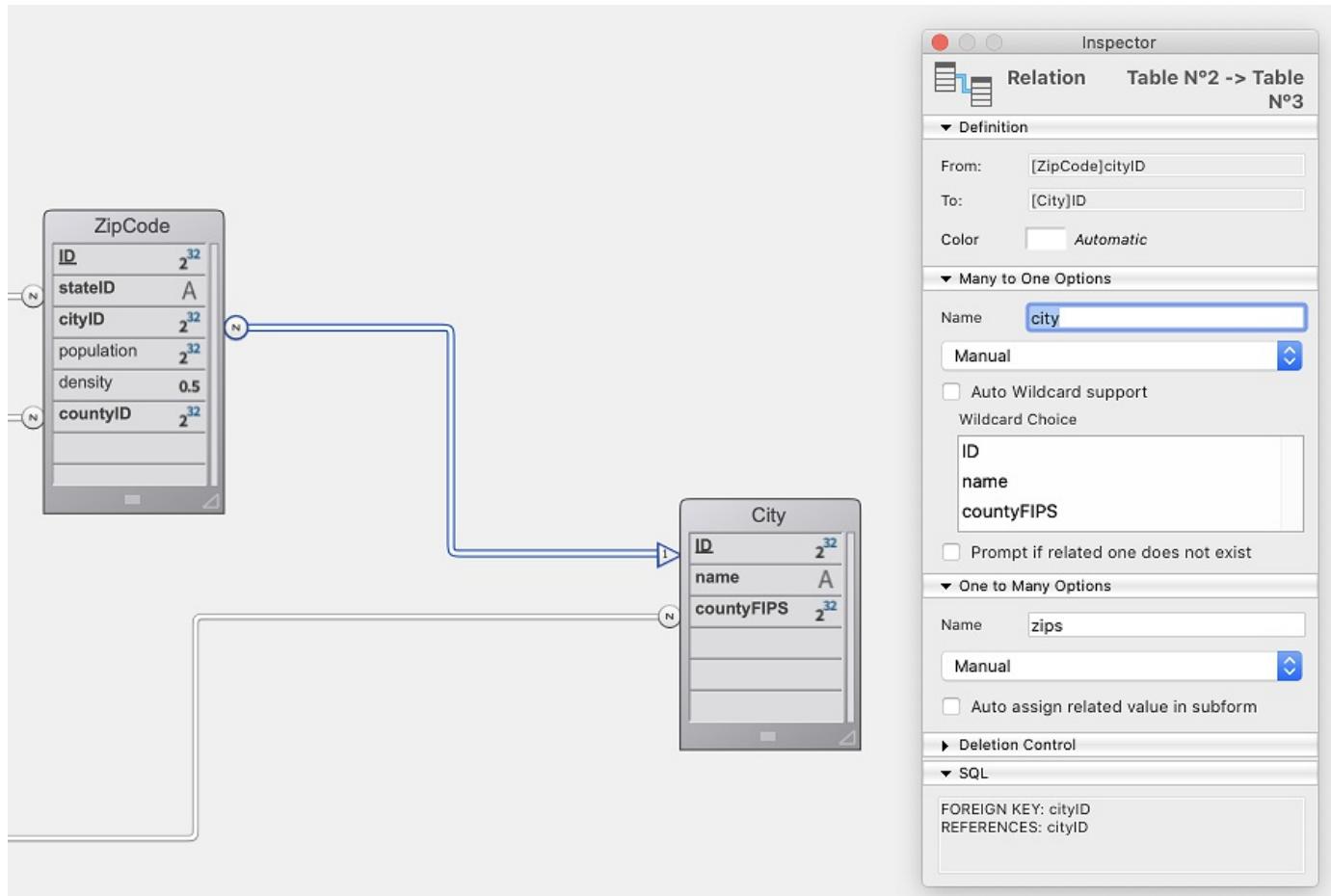
Si la petición envía una selección de entidades modificada al servidor, la función del modelo de datos ORDA llamada se ejecutará automáticamente en el servidor con la selección de entidades modificada.

Propiedades	Tipo	Descripción
Atributos de la entidad	mixto	Opcional - Valores a modificar
__DATASET	Cadena	Obligatorio - entitySetID (UUID) de la selección de entidades
__ENTITIES	Booleano	Obligatorio - True para indicar al servidor que el parámetro es una selección de entidades

Ver ejemplo para [recibir una selección de entidades](#).

## Ejemplos de peticiones

Esta base de datos se expone como un almacén de datos remoto en localhost (puerto 8111):



## Utilizar una función de clase de datastore

La clase de `DataStore US_Cities` ofrece una API:

```
// DataStore class

Class extends DataStoreImplementation

exposed Function getName()
$0:="US cities and zip codes manager"
```

A continuación, puede ejecutar esta petición:

```
POST 127.0.0.1:8111/rest/$catalog/getName
```

Resultado

```
{  
  "result": "US cities and zip codes manager"  
}
```

## Utilizar una función de clase de dataclass

La clase de Dataclass `City` ofrece una PI que devuelve una entidad de ciudad a partir del nombre pasado en parámetro:

```
// City class  
  
Class extends DataClass  
  
exposed Function getCity()  
  var $0 : cs.CityEntity  
  var $1,$nameParam : text  
  $nameParam:=$1  
  $0:=This.query("name = :1";$nameParam).first()
```

A continuación, puede ejecutar esta petición:

```
POST 127.0.0.1:8111/rest/City/getCity
```

Petición: ["Aguada"]

Resultado

El resultado es una entidad:

```
{  
  "__entityModel": "City",  
  "__DATACLASS": "City",  
  "__KEY": "1",  
  "__TIMESTAMP": "2020-03-09T08:03:19.923Z",  
  "__STAMP": 1,  
  "ID": 1,  
  "name": "Aguada",  
  "countyFIPS": 72003,  
  "county": {  
    "__deferred": {  
      "uri": "/rest/County(72003)",  
      "__KEY": "72003"  
    }  
  },  
  "zips": {  
    "__deferred": {  
      "uri": "/rest/City(1)/zips?$expand=zips"  
    }  
  }  
}
```

## Utilizar una función de clase de una entidad

La clase de entidad `CityEntity` ofrece una API:

```
// CityEntity class

Class extends Entity

exposed Function getPopulation()
$0:=This.zips.sum("population")
```

A continuación, puede ejecutar esta petición:

POST 127.0.0.1:8111/rest/City(2)/getPopulation

Resultado

```
{
    "result": 48814
}
```

## Utilizar una función clase entitySelection

La clase de selección de entidad `CityEntity` ofrece una API:

```
// CitySelection class

Class extends EntitySelection

exposed Function getPopulation()
$0:=This.zips.sum("population")
```

A continuación, puede ejecutar esta petición:

POST 127.0.0.1:8111/rest/City/getPopulation/?\$filter="ID<3"

Resultado

```
{
    "result": 87256
}
```

## Utilizar una función de clase de selección de entidades y un conjunto de entidades

La clase `StudentsSelection` tiene una función `getAgeAverage`:

```
// StudentsSelection Class

Class extends EntitySelection

exposed Function getAgeAverage
    C_LONGINT($sum;$0)
    C_OBJECT($s)

    $sum:=0
    For each ($s;This)
        $sum:=$sum+$s.age()
    End for each
    $0:=$sum/This.length
```

Una vez que haya creado un conjunto de entidades, puede ejecutar esta petición:

```
POST 127.0.0.1:8044/rest/Students/getAgeAverage/$entityset/17E83633FFB54ECDBF947E5C620BB532
```

Resultado

```
{  
    "result": 34  
}
```

Utilizar una función de clase de selección de entidades y unorderBy

La clase `StudentsSelection` tiene una función `getLastSummary`:

```
// StudentsSelection Class  
  
Class extends EntitySelection  
  
exposed Function getLastSummary  
    C_TEXT($0)  
    C_OBJECT($last)  
  
    $last:=This.last()  
    $0:=$last.firstname + " - +"$last.lastname+ " is ... "+String($last.age())
```

A continuación, puede ejecutar esta petición:

```
POST 127.0.0.1:8044/rest/Students/getLastSummary/$entityset/?$filter="lastname=b@"&$orderby="lastname"
```

Resultado

```
{  
    "result": "Wilbert - Bull is ... 21"  
}
```

Utilizar una entidad que se creará en el servidor

La clase de Dataclass `Students` tiene la función `pushData()` que recibe una entidad que contiene los datos del cliente. El método `checkData()` efectúa algunos controles. Si son válidos, la entidad se guarda y se devuelve.

```

// Students Class

Class extends DataClass

exposed Function pushData
    var $1, $entity, $status, $0 : Object

    $entity:=$1

    $status:=checkData($entity) // $status es un objeto con una propiedad booleana "success"

    $0:=$status

    If ($status.success)
        $status:=$entity.save()
        If ($status.success)
            $0:=$entity
    End if
End if

```

Lance esta petición:

POST <http://127.0.0.1:8044/rest/Students/pushData>

Cuerpo de la petición:

```
{
  "__DATACLASS": "Students",
  "__ENTITY": true,
  "firstname": "Ann",
  "lastname": "Brown"
}]
```

Como ninguna `__KEY` es dada, una nueva entidad Students está cargada en el servidor con los atributos del cliente.  
Como la función `pushData()` ejecuta una acción `save()`, la nueva entidad es creada.

Resultado

```
{
  "__entityModel": "Students",
  "__DATACLASS": "Students",
  "__KEY": "55",
  "__TIMESTAMP": "2020-06-16T10:54:41.805Z",
  "__STAMP": 1,
  "ID": 55,
  "firstname": "Ann",
  "lastname": "BROWN",
  "schoolID": null,
  "school": null
}
```

Utilizar una entidad a actualizar en el servidor

Igual que el anterior pero con el atributo `__KEY`

Lance esta petición:

POST: <http://127.0.0.1:8044/rest/Students/pushData>

Cuerpo de la petición:

```
[{  
    "__DATACLASS": "Students",  
    "__ENTITY": true,  
    "lastname": "Brownie",  
    "__KEY": 55  
}]
```

Como `__KEY` es dada, la entidad Students está cargada con llave primaria 55 con el valor lastname recibido por el cliente. Como la función ejecuta una acción `save()`, la nueva entidad es actualizada.

Resultado

```
{  
    "__entityModel": "Students",  
    "__DATACLASS": "Students",  
    "__KEY": "55",  
    "__TIMESTAMP": "2020-06-16T11:10:21.679Z",  
    "__STAMP": 3,  
    "ID": 55,  
    "firstname": "Ann",  
    "lastname": "BROWNIE",  
    "schoolID": null,  
    "school": null  
}
```

## Crear una entidad con una entidad relacionada

En este ejemplo, creamos una nueva entidad Students con la entidad Schools con la llave primaria 2.

Lance esta petición:

POST: <http://127.0.0.1:8044/rest/Students/pushData>

Cuerpo de la petición:

```
[{  
    "__DATACLASS": "Students",  
    "__ENTITY": true,  
    "firstname": "John",  
    "lastname": "Smith",  
    "school": {"__KEY": 2}  
}]
```

Resultado

```
{
    "__entityModel": "Students",
    "__DATACLASS": "Students",
    "__KEY": "56",
    "__TIMESTAMP": "2020-06-16T11:16:47.601Z",
    "__STAMP": 1,
    "ID": 56,
    "firstname": "John",
    "lastname": "SMITH",
    "schoolID": 2,
    "school": {
        "__deferred": {
            "uri": "/rest/Schools(2)",
            "__KEY": "2"
        }
    }
}
```

## Actualizar una entidad con una entidad relacionada

En este ejemplo, asociamos una escuela existente a una entidad Students. La clase `StudentsEntity` tiene una API:

```
// StudentsEntity class

Class extends Entity

exposed Function putToSchool()
    var $1, $school , $0, $status : Object

    // $1 es una entidad Schools
    $school:=$1
        // Asocia la entidad relacionada "school" con la entidad actual "Students"
    This.school:=$school

    $status:=This.save()

    $0:=$status
```

Se ejecuta esta petición, llamada en una entidad Students: POST

`http://127.0.0.1:8044/rest/Students(1)/putToSchool` Cuerpo de la petición:

```
[{
    "__DATACLASS":"Schools",
    "__ENTITY":true,
    "__KEY":2
}]
```

## Resultado

```
{
    "result": {
        "success": true
    }
}
```

## Recibir una selección de entidades como parámetro

En la clase de Dataclass `Students`, la función `setFinalExam()` actualiza una selección de entidad recibida (\$1). En realidad, actualiza el atributo `finalExam` con el valor recibido (\$2). Devuelve las llaves primarias de las entidades actualizadas.

```
// Students class

Class extends DataClass

exposed Function setFinalExam()

var $1, $es, $student, $status : Object
var $2, $examResult : Text

var $keys, $0 : Collection

//Entity selection
$es:=$1

$examResult:=$2

$keys:=New collection()

//Bucle en la selección de entidades
For each ($student;$es)
    $student.finalExam:=$examResult
    $status:=$student.save()
    If ($status.success)
        $keys.push($student.ID)
    End if
End for each

$0:=$keys
```

Un conjunto de entidades se crea primero con esta petición:

```
http://127.0.0.1:8044/rest/Students/?$filter="ID<3"&$method=entityset
```

A continuación, puede ejecutar esta petición:

```
POST http://127.0.0.1:8044/rest/Students/setFinalExam
```

Cuerpo de la petición:

```
[
{
  "__ENTITIES":true,
  "__DATASET":"9B9C053A111E4A288E9C1E48965FE671"
},
"Passed"
]
```

Resultado

Se han actualizado las entidades con llaves primarias 1 y 2.

```
{  
  "result": [  
    1,  
    2  
  ]  
}
```

## Utilizar una selección de entidades actualizada en el cliente

Utilizando la función `getAgeAverage()` [definida anteriormente](#).

```
var $remoteDS, $newStudent, $students : Object  
var $ageAverage : Integer  
  
$remoteDS:=Open datastore(New object("hostname","127.0.0.1:8044");"students")  
  
// $newStudent es una entidad "student" a procesar  
$newStudent:=...  
$students:=$remoteDS.Students.query("school.name = :1";"Math school")  
// Hemos añadido una entidad a la selección de entidades $students en el cliente  
$students.add($newStudent)  
  
// Llamamos a una función en la clase StudentsSelection que devuelve la edad media de los estudiantes en  
// La función se utiliza en el servidor en la selección de la entidad $students actualizada, que incluye  
$ageAverage:=$students.getAgeAverage()
```

# Acerca de las peticiones REST

Se soportan las siguientes estructuras para las peticiones REST:

URI	Resource (Input)	/? or &{filter} (Output)
http://{servername}:{port}/rest/	{dataClass}	\$filter, \$attributes, \$skip, \$method=.....
	{dataClass}/\$entityset/{entitySetID}	\$method=...
	{dataClass}({key})	\$attributes
	{dataClass}:{attribute}(value)	

Mientras que todas las solicitudes REST deben contener los parámetros URI y Resource, los filtros de salida (que filtran los datos devueltos) son opcionales.

Como en todos los URI, el primer parámetro está delimitado por un "?" y todos los siguientes por un "&". Por ejemplo:

```
GET /rest/Person/?$filter="lastName!=Jones"&$method=entityset&$timeout=600
```

Puede colocar todos los valores entre comillas en caso de ambigüedad. Por ejemplo, en nuestro ejemplo anterior, podríamos haber puesto el valor del apellido entre comillas simples: "lastName!='Jones'".

Los parámetros le permiten manipular los datos de las clases de datos en su proyecto 4D. Además de recuperar datos mediante los métodos HTTP `GET`, también se pueden añadir, actualizar y eliminar entidades de una clase de datos utilizando los métodos HTTP `POST`.

Si desea que los datos se devuelvan en un array en lugar de en JSON, utilice el parámetro `$asArray`.

## Estado y respuesta REST

Con cada petición REST, el servidor devuelve el estado y una respuesta (con o sin error).

### Estado de la petición

Con cada solicitud REST, se obtiene el estado junto con la respuesta. A continuación se presentan algunos de los estados que pueden surgir:

Estado	Descripción
0	Solicitud no procesada (el servidor podría no estar iniciado).
200 OK	Petición procesada sin error.
401 Unauthorized	Error de permisos (compruebe los permisos del usuario).
402 No session	Se ha alcanzado el número máximo de sesiones.
404 Not Found	La clase de datos no es accesible vía REST o el conjunto de entidades no existe.
500 Internal Server Error	Error al procesar la solicitud REST.

### Respuesta

La respuesta (en formato JSON) varía en función de la petición.

Si se produce un error, se enviará junto con la respuesta del servidor o será la respuesta del servidor.



# \$catalog

El catálogo describe todas las clases de datos y atributos disponibles en el almacén de datos.

## Sintaxis disponible

Sintaxis	Ejemplo	Descripción
\$catalog	/\$catalog	Devuelve una lista de las clases de datos de su proyecto junto con dos URIs
\$catalog/\$all	/\$catalog/\$all	Devuelve información sobre todas las clases de datos del proyecto y sus atributos
\$catalog/{dataClass}	/\$catalog/Employee	Devuelve información sobre una clase de datos y sus atributos

## \$catalog

Devuelve una lista de las clases de datos de su proyecto junto con dos URI: una para acceder a la información sobre su estructura y otra para recuperar los datos de la clase de datos

### Descripción

Cuando se llama a `$catalog`, se devuelve una lista de las clases de datos junto con dos URI para cada clase de datos en el almacén de datos de su proyecto.

En esta lista sólo se muestran las clases de datos expuestas para el almacén de datos de su proyecto. Para más información, consulte la sección [Exponer tablas y campos](#).

A continuación se describen las propiedades devueltas para cada clase de datos en el almacén de datos de su proyecto:

Propiedad	Tipo	Descripción
name	Cadena	Nombre de la dataclass.
uri	Cadena	Un URI que permite obtener información sobre la clase de datos y sus atributos.
dataURI	Cadena	Un URI que permite ver los datos en la clase de datos.

### Ejemplo

```
GET /rest/$catalog
```

Resultado:

```
{
  dataClasses: [
    {
      name: "Company",
      uri: "http://127.0.0.1:8081/rest/$catalog/Company",
      dataURI: "http://127.0.0.1:8081/rest/Company"
    },
    {
      name: "Employee",
      uri: "http://127.0.0.1:8081/rest/$catalog/Employee",
      dataURI: "http://127.0.0.1:8081/rest/Employee"
    }
  ]
}
```

## \$catalog/\$all

Devuelve información sobre todas las clases de datos del proyecto y sus atributos

### Descripción

Llamando `$catalog/$all` puede recibir información detallada sobre los atributos de cada una de las clases de datos del modelo activo del proyecto.

Para más información sobre lo que se devuelve para cada clase de datos y sus atributos, utilice `$catalog/{dataClass}`.

### Ejemplo

`GET /rest/$catalog/$all`

Resultado:

```
{
  "dataClasses": [
    {
      "name": "Company",
      "className": "Company",
      "collectionName": "CompanySelection",
      "tableNumber": 2,
      "scope": "public",
      "dataURI": "/rest/Company",
      "attributes": [
        {
          "name": "ID",
          "kind": "storage",
          "fieldPos": 1,
          "scope": "public",
          "indexed": true,
          "type": "long",
          "identifying": true
        },
        {
          "name": "name",
          "kind": "storage",
          "fieldPos": 2,
          "scope": "public",
          "type": "string"
        },
        {
          "name": "revenues",
          "kind": "storage",
          "fieldPos": 3,
          "scope": "public",
          "type": "double"
        }
      ]
    }
  ]
}
```

```
        "kind": "storage",
        "fieldPos": 3,
        "scope": "public",
        "type": "number"
    },
    {
        "name": "staff",
        "kind": "relatedEntities",
        "fieldPos": 4,
        "scope": "public",
        "type": "EmployeeSelection",
        "reversePath": true,
        "path": "employer"
    },
    {
        "name": "url",
        "kind": "storage",
        "scope": "public",
        "type": "string"
    }
],
"key": [
    {
        "name": "ID"
    }
]
},
{
    "name": "Employee",
    "className": "Employee",
    "collectionName": "EmployeeSelection",
    "tableNumber": 1,
    "scope": "public",
    "dataURI": "/rest/Employee",
    "attributes": [
        {
            "name": "ID",
            "kind": "storage",
            "scope": "public",
            "indexed": true,
            "type": "long",
            "identifying": true
        },
        {
            "name": "firstname",
            "kind": "storage",
            "scope": "public",
            "type": "string"
        },
        {
            "name": "lastname",
            "kind": "storage",
            "scope": "public",
            "type": "string"
        },
        {
            "name": "employer",
            "kind": "relatedEntity",
            "scope": "public",
            "type": "Company",
            "path": "Company"
        }
    ],
    "key": [
        {
            "name": "ID"
        }
    ]
}
```

```
        "name": "ID"
    }
}
]
}
```

## \$catalog/{dataClass}

Devuelve información sobre una clase de datos y sus atributos

### Descripción

La llamada de `$catalog/{dataClass}` para una clase de datos específica devolverá la siguiente información sobre la clase de datos y los atributos que contiene. Si quiere recuperar esta información para todas las clases de datos del almacén de datos de su proyecto, utilice `$catalog/$all`.

La información que recupera se refiere a lo siguiente:

- Dataclass
- Atributo(s)
- Método(s) si lo hay
- Llave primaria

### DataClass

Las siguientes propiedades se devuelven para una clase de datos expuesta:

Propiedad	Tipo	Descripción
name	Cadena	Nombre de la dataclass
collectionName	Cadena	Nombre de una selección de entidades en la clase de datos
tableNumber	Número	Número de tabla en la base 4D
scope	Cadena	Alcance de la clase de datos (tenga en cuenta que sólo se muestran las clases de datos cuyo Alcance es público)
dataURI	Cadena	Un URI a los datos de la clase de datos

### Atributo(s)

Aquí están las propiedades de cada atributo expuesto que se devuelven:

Propiedad	Tipo	Descripción
name	Cadena	El nombre del atributo.
kind	Cadena	Tipo de atributo (almacenamiento o entidad relacionada).
fieldPos	Número	Posición del campo en la tabla de la base).
scope	Cadena	Alcance del atributo (sólo aparecerán los atributos cuyo alcance sea Público).
indexed	Cadena	Si se seleccionó algún tipo de índice, esta propiedad devolverá true. En caso contrario, esta propiedad no aparece.
type	Cadena	Tipo de atributo (booleano, blob, byte, fecha, duración, imagen, long, long64, número, cadena, uuid o palabra) o la clase de datos para un atributo de relación N->1.
identifying	Booleano	Esta propiedad devuelve True si el atributo es la llave primaria. En caso contrario, esta propiedad no aparece.
path	Cadena	Nombre de la clase de datos para un atributo relatedEntity, o nombre de la relación para un atributo relatedEntities.
foreignKey	Cadena	Para un atributo relatedEntity, nombre del atributo relacionado.
inverseName	Cadena	Nombre de la relación opuesta para un atributo relatedEntity o relateEntities.

## Llave primaria

El objeto llave devuelve el nombre del atributo name definido como llave primaria para la clase de datos.

## Ejemplo

Puede recuperar la información relativa a una clase de datos específica.

```
GET /rest/$catalog/Employee
```

Resultado:

```
{
  name: "Employee",
  className: "Employee",
  collectionName: "EmployeeCollection",
  scope: "public",
  dataURI: "http://127.0.0.1:8081/rest/Employee",
  defaultTopSize: 20,
  extraProperties: {
    panelColor: "#76923C",
    __CDATA: "\n\n\t\t\n",
    panel: {
      isOpen: "true",
      pathVisible: "true",
      __CDATA: "\n\n\t\t\t\n",
      position: {
        X: "394",
        Y: "42"
      }
    }
  },
  attributes: [
    {
      name: "ID",
      kind: "storage",
      scope: "public",
      indexed: true,
      type: "long",
      identifying: true
    }
  ]
}
```

```
,  
{  
    name: "firstName",  
    kind: "storage",  
    scope: "public",  
    type: "string"  
},  
{  
    name: "lastName",  
    kind: "storage",  
    scope: "public",  
    type: "string"  
},  
{  
    name: "fullName",  
    kind: "calculated",  
    scope: "public",  
    type: "string",  
    readOnly: true  
},  
{  
    name: "salary",  
    kind: "storage",  
    scope: "public",  
    type: "number",  
    defaultFormat: {  
        format: "$###,###.00"  
    }  
},  
{  
    name: "photo",  
    kind: "storage",  
    scope: "public",  
    type: "image"  
},  
{  
    name: "employer",  
    kind: "relatedEntity",  
    scope: "public",  
    type: "Company",  
    path: "Company"  
},  
{  
    name: "employerName",  
    kind: "alias",  
    scope: "public",  
  
    type: "string",  
    path: "employer.name",  
    readOnly: true  
},  
{  
    name: "description",  
    kind: "storage",  
    scope: "public",  
    type: "string",  
    multiLine: true  
},  
,  
key: [  
    {  
        name: "ID"  
    }  
]  
}
```



# \$directory

El directorio gestiona el acceso de los usuarios a través de peticiones REST.

## \$directory/login

Abre una sesión REST en su aplicación 4D y conecta al usuario.

### Descripción

Utilice `$directory/login` para abrir una sesión en su aplicación 4D a través de REST y conecte un usuario. También puede modificar el tiempo de espera por defecto de la sesión 4D.

Todos los parámetros deben pasarse en encabezados de un método POST:

Llave de encabezado	Valor del encabezado
username-4D	Usuario - No obligatorio
password-4D	Contraseña - No obligatorio
hashed-password-4D	Contraseña hashed - No obligatorio
session-4D-length	Tiempo de inactividad de la sesión (minutos). No puede ser inferior a 60 - No es obligatorio

### Ejemplo

```
C_TEXT($response;$body_t)
ARRAY TEXT($hKey;3)
ARRAY TEXT($hValues;3)
$hKey{1}:="username-4D"
$hKey{2}:="hashed-password-4D"
$hKey{3}:="session-4D-length"
$hValues{1}:="john"
$hValues{2}:=Generate digest("123";4D digest)
$hValues{3}:=120
$httpStatus:=HTTP Request(HTTP POST method;"app.example.com:9000/rest/$directory/login";$body_t;$respon
```

Resultado:

Si la conexión fue exitosa, el resultado será:

```
{
  "result": true
}
```

De lo contrario, la respuesta será:

```
{
  "result": false
}
```



# \$info

Devuelve información sobre los conjuntos de entidades almacenados actualmente en la caché de 4D Server, así como las sesiones usuario

## Descripción

Cuando llama a esta petición para su proyecto, se recupera la información en las propiedades siguientes:

Propiedad	Tipo	Descripción
cacheSize	Número	Tamaño de la caché del servidor 4D.
usedCache	Número	Cuánto se ha utilizado de la caché de 4D Server.
entitySetCount	Número	Número de selecciones de entidades.
entitySet	Collection	Una colección en la que cada objeto contiene información sobre cada selección de entidades.
ProgressInfo	Collection	Una colección que contiene información sobre el indicador de progreso.
sessionInfo	Collection	Una colección en la que cada objeto contiene información sobre cada sesión usuario.

## entitySet

Para cada selección de entidad almacenada actualmente en la caché de 4D Server, se devuelve la siguiente información:

Propiedad	Tipo	Descripción
id	Cadena	Un UUID que hace referencia al conjunto de entidades.
dataClass	Cadena	Nombre de la dataclass.
selectionSize	Número	Número de entidades en la selección de entidades.
sorted	Booleano	Devuelve true si el conjunto fue ordenado ( utilizando <code>\$orderby</code> ) o false si no está ordenado.
refreshed	Fecha	Cuando se creó el conjunto de entidades o la última vez que se utilizó.
expires	Fecha	Cuándo expirará el conjunto de entidades (esta fecha/hora cambia cada vez que se actualiza el conjunto de entidades). La diferencia entre refrescado y vencido es el tiempo de espera de un conjunto de entidades. Este valor es de dos horas por defecto o lo que hayas definido utilizando <code>\$timeout</code> .

Para obtener información sobre cómo crear una selección de entidades, consulte `$method=entityset` . Si desea eliminar la selección de entidades de la caché de 4D Server, utilice `$method=release` .

4D también crea sus propias selecciones de entidades con fines de optimización, por lo que las que se crean con `$method=entityset` no son las únicas que se devuelven. **IMPORTANT** If your project is in Controlled Admin Access Mode, you must first log into the project as a user in the Admin group.

## sessionInfo

Para cada sesión de usuario, se devuelve la siguiente información en la colección `sessionInfo`:

Propiedad	Tipo	Descripción
sessionID	Cadena	Un UUID que referencia la sesión.
userName	Cadena	El nombre del usuario que ejecuta la sesión.
lifeTime	Número	La duración de una sesión usuario en segundos (3600 por defecto).
expiration	Fecha	La fecha y la hora actuales de caducidad de la sesión de usuario.

## Ejemplo

Recupera la información sobre los conjuntos de entidades almacenados actualmente en la caché de 4D Server, así como las sesiones usuario:

```
GET /rest/$info
```

Resultado:

```

{
cacheSize: 209715200,
usedCache: 3136000,
entitySetCount: 4,
entitySet: [
{
id: "1418741678864021B56F8C6D77F2FC06",
tableName: "Company",
selectionSize: 1,
sorted: false,
refreshed: "2011-11-18T10:30:30Z",
expires: "2011-11-18T10:35:30Z"
},
{
id: "CAD79E5BF339462E85DA613754C05CC0",
tableName: "People",
selectionSize: 49,
sorted: true,
refreshed: "2011-11-18T10:28:43Z",
expires: "2011-11-18T10:38:43Z"
},
{
id: "F4514C59D6B642099764C15D2BF51624",
tableName: "People",
selectionSize: 37,
sorted: false,
refreshed: "2011-11-18T10:24:24Z",
expires: "2011-11-18T12:24:24Z"
}
],
ProgressInfo: [
{
UserInfo: "flushProgressIndicator",
sessions: 0,
percent: 0
},
{
UserInfo: "indexProgressIndicator",
sessions: 0,
percent: 0
}
],
sessionInfo: [
{
sessionID: "6657ABBCEE7C3B4089C20D8995851E30",
userID: "36713176D42DB045B01B8E650E8FA9C6",
userName: "james",
lifeTime: 3600,
expiration: "2013-04-22T12:45:08Z"
},
{
sessionID: "A85F253EDE90CA458940337BE2939F6F",
userID: "00000000000000000000000000000000",
userName: "default guest",
lifeTime: 3600,
expiration: "2013-04-23T10:30:25Z"
}
]
}

```

La información del indicador de progreso que aparece después de las selecciones de entidades es utilizada internamente por 4D.



# \$upload

Devuelve un ID del archivo subido al servidor

## Descripción

Publique esta petición cuando tenga un archivo que quiera subir al Servidor. Si tiene una imagen, pase `$rawPict=true`. Para todos los demás archivos, se pasa `$binary=true`.

Puede modificar el tiempo de espera, que por defecto es de 120 segundos, pasando un valor al parámetro `$timeout`.

## Escenario de carga

Imagine que quiere subir una imagen para actualizar el atributo imagen de una entidad.

Para cargar una imagen (o cualquier archivo binario), primero debe seleccionar el archivo desde la aplicación cliente. El archivo en sí debe pasarse en el cuerpo de la petición.

A continuación, se sube la imagen seleccionada a 4D Server mediante una petición como:

```
POST /rest/$upload?$rawPict=true
```

Como resultado, el servidor devuelve un ID que identifica el archivo:

Respuesta:

```
{ "ID": "D507BC03E613487E9B4C2F6A0512FE50" }
```

Después, se utiliza este ID para añadirlo a un atributo utilizando `$method=update` para añadir la imagen a una entidad. La petición se ve así:

```
POST /rest/Employee/?$method=update
```

Datos POST:

```
{
  KEY: "12",
  STAMP: 4,
  photo: { "ID": "D507BC03E613487E9B4C2F6A0512FE50" }
}
```

Respuesta:

Se devuelve la entidad modificada:

```
{
    "__KEY": "12",
    "__STAMP": 5,
    "uri": "http://127.0.0.1:8081/rest/Employee(12)",
    "ID": 12,
    "firstName": "John",
    "firstName": "Smith",
    "photo":
    {
        "__deferred":
        {
            "uri": "/rest/Employee(12)/photo?$imageformat=best&$version=1&$expand=photo",
            "image": true
        }
    }
},}
```

## Ejemplo con un cliente 4D HTTP

El siguiente ejemplo muestra cómo subir un archivo .pdf al servidor utilizando el cliente 4D HTTP.

```
var $params : Text
var $response : Object
var $result : Integer
var $blob : Blob

ARRAY TEXT($headerNames; 1)
ARRAY TEXT($headerValues; 1)

$url:="localhost:80/rest/$upload?$binary=true" //preparar une petición REST

$headerNames{1}:="Content-Type"
$headerValues{1}:="application/octet-stream"

DOCUMENT TO BLOB("c:\\invoices\\inv003.pdf"; $blob) //Cargar el binario

//Ejecuta la primera petición POST para subir el archivo
$result:=HTTP Request(HTTP POST method; $url; $blob; $response; $headerNames; $headerValues)

If ($result=200)
    var $data : Object
    $data:=New object
    $data.__KEY:="3"
    $data.__STAMP:="3"
    $data.pdf:=New object("ID"; String($response.ID))

    $url:="localhost:80/rest/Invoices?$method=update" //segunda petición para actualizar la entidad

    $headerNames{1}:="Content-Type"
    $headerValues{1}:="application/json"

    $result:=HTTP Request(HTTP POST method; $url; $data; $response; $headerNames; $headerValues)
Else
    ALERT(String($result)+" Error")
End if
```

# {dataClass}

Los nombres de dataclass pueden utilizarse directamente en las peticiones REST para trabajar con entidades, selecciones de entidades o funciones de clase de la dataclass.

## Sintaxis disponible

Sintaxis	Ejemplo	Descripción
{dataClass}	/Employee	Devuelve todos los datos (por defecto las 100 primeras entidades) de la clase de datos
{dataClass}({key})	/Employee(22)	Devuelve los datos de la entidad específica definida por la llave primaria de la clase de datos
{dataClass}:{attribute}(value)	/Employee:firstName(John)	Devuelve los datos de una entidad en la que está definido el valor del atributo
{dataClass}/{DataClassClassFunction}	/City/getCity	Ejecuta una función de clase de una dataclass
{dataClass}({EntitySelectionClassFunction})	/City/getPopulation/?\$filter="ID<3"	Ejecuta una función de clase de una selección de entidades
{dataClass}({key})/{EntityClassFunction}	City(2)/getPopulation	Ejecuta una función de clase de una entidad

Las llamadas a las funciones se detallan en la sección [Llamar las funciones de la clase ORDA](#).

# {dataClass}

Devuelve todos los datos (por defecto las 100 primeras entidades) para una clase de datos específica ( *por ejemplo, Company* )

## Descripción

Cuando se llama a este parámetro en la petición REST, se devuelven las 100 primeras entidades, a menos que se haya especificado un valor con `$top/$limit`.

A continuación se describen los datos devueltos:

Propiedad	Tipo	Descripción
__entityModel	Cadena	Nombre de la dataclass.
__COUNT	Número	Número de entidades en la clase de datos.
__SENT	Número	Número de entidades enviadas por la petición REST. Este número puede ser el número total de entidades si es menor que el valor definido por <code>\$top/\$limit</code> .
__FIRST	Número	Número de entidad en la que comienza la selección. O bien 0 por defecto o el valor definido por <code>\$skip</code> .
__ENTITIES	Collection	Esta colección de objetos contiene un objeto para cada entidad con todos sus atributos. Todos los atributos relacionales se devuelven como objetos con una URI para obtener información sobre el padre.

Cada entidad contiene las siguientes propiedades:

Propiedad	Tipo	Descripción
__KEY	Cadena	Valor de la llave primaria definida para la clase de datos.
__TIMESTAMP	Fecha	Marca de tiempo de la última modificación de la entidad
__STAMP	Número	Sello interno que se necesita cuando se modifica alguno de los valores de la entidad al utilizar <code>\$method=update</code> .

Si quiere especificar qué atributos quiere devolver, defínalos utilizando la siguiente sintaxis `{attribute1, attribute2, ...}`. Por ejemplo:

```
GET /rest/Company/name,address
```

## Ejemplo

Devuelve todos los datos de una clase de datos específica.

```
GET /rest/Company
```

Resultado:

```
{
    "__entityModel": "Company",
    "__GlobalStamp": 51,
    "__COUNT": 250,
    "__SENT": 100,
    "__FIRST": 0,
    "__ENTITIES": [
        {
            "__KEY": "1",
            "__TIMESTAMP": "2020-04-10T10:44:49.927Z",
            "__STAMP": 1,
            "ID": 1,
            "name": "Adobe",
            "address": null,
            "city": "San Jose",
            "country": "USA",
            "revenues": 500000,
            "staff": {
                "__deferred": {
                    "uri": "http://127.0.0.1:8081/rest/Company(1)/staff?$expand=staff"
                }
            }
        },
        {
            "__KEY": "2",
            "__TIMESTAMP": "2018-04-25T14:42:18.351Z",
            "__STAMP": 1,
            "ID": 2,
            "name": "Apple",
            "address": null,
            "city": "Cupertino",
            "country": "USA",
            "revenues": 890000,
            "staff": {
                "__deferred": {
                    "uri": "http://127.0.0.1:8081/rest/Company(2)/staff?$expand=staff"
                }
            }
        },
        {
            "__KEY": "3",
            "__TIMESTAMP": "2018-04-25T00:00:10.001Z"
        }
    ]
}
```

```

    "__TIMESTAMP": "2018-04-23T09:03:49.021Z",
    "__STAMP": 2,
    "ID": 3,
    "name": "4D",
    "address": null,
    "city": "Clichy",
    "country": "France",
    "revenues": 700000,
    "staff": {
        "__deferred": {
            "uri": "http://127.0.0.1:8081/rest/Company(3)/staff?$expand=staff"
        }
    }
},
{
    "__KEY": "4",
    "__TIMESTAMP": "2018-03-28T14:38:07.430Z",
    "__STAMP": 1,
    "ID": 4,
    "name": "Microsoft",
    "address": null,
    "city": "Seattle",
    "country": "USA",
    "revenues": 650000,
    "staff": {
        "__deferred": {
            "uri": "http://127.0.0.1:8081/rest/Company(4)/staff?$expand=staff"
        }
    }
}
....//more entities here
]
}

```

## {dataClass}({key})

Devuelve los datos de la entidad específica definida por la llave primaria de la clase de datos, *p. ej.*, `Company(22)` o `Company("IT0911AB2200")`

### Descripción

Pasando la clase de datos y una llave, se puede recuperar toda la información pública de esa entidad. Pasando la clase de datos y una llave, se puede recuperar toda la información pública de esa entidad. Para más información sobre la definición de una llave primaria, consulte la sección [Modifying the Primary Key](#) en el Editor del modelo de datos.

Para más información sobre los datos devueltos, consulte [{DataStoreClass}](#).

Si quiere especificar qué atributos quiere devolver, defínalos utilizando la siguiente sintaxis `{attribute1, attribute2, ...}`. Por ejemplo:

```
GET /rest/Company(1)/name,address
```

Si desea expandir un atributo de relación utilizando `$expand`, lo hará especificándolo como se muestra a continuación:

```
GET /rest/Company(1)/name,address,staff?$expand=staff
```

### Ejemplo

La siguiente petición devuelve todos los datos públicos de la clase de datos `Company` cuya llave es 1.

```
GET /rest/Company(1)
```

Resultado:

```
{  
    "__entityModel": "Company",  
    "__KEY": "1",  
    "__TIMESTAMP": "2020-04-10T10:44:49.927Z",  
    "__STAMP": 2,  
    "ID": 1,  
    "name": "Apple",  
    "address": Infinite Loop,  
    "city": "Cupertino",  
    "country": "USA",  
    "url": http://www.apple.com,  
    "revenues": 500000,  
    "staff": {  
        "__deferred": {  
            "uri": "http://127.0.0.1:8081/rest/Company(1)/staff?$expand=staff"  
        }  
    }  
}
```

## {dataClass}:{attribute}(value)

Devuelve los datos de una entidad en la que está definido el valor del atributo

### Descripción

Pasando la *clase de datos* y un *atributo* junto con un valor, se puede recuperar toda la información pública de esa entidad. El valor es un valor único para el atributo, pero no es la llave primaria.

```
GET /rest/Company:companyCode(Acme001)
```

Si quiere especificar qué atributos quiere devolver, defínalos utilizando la siguiente sintaxis `{attribute1, attribute2, ...}`. Por ejemplo:

```
GET /rest/Company:companyCode(Acme001)/name,address
```

Si desea utilizar un atributo relacional utilizando [/docs/Rx/es/REST/attributes.html](#), lo hará especificándolo como se muestra a continuación:

```
GET /rest/Company:companyCode(Acme001)?$attributes=name,address,staff.name
```

### Ejemplo

La siguiente petición devuelve todos los datos públicos del empleado llamado "Jones".

```
GET /rest/Employee:lastname(Jones)
```

# \$asArray

Devuelve el resultado de una petición en un array (es decir, una colección) en lugar de un objeto JSON.

## Descripción

Si desea obtener la respuesta en un array, sólo tiene que añadir `$asArray` a su petición REST (e.g., `$asArray=true`).

## Ejemplo

Aquí hay un ejemplo de cómo recibir la respuesta en un array.

```
GET /rest/Company/?$filter="name begin a"&$top=3&$asArray=true
```

Respuesta:

```
[  
  {  
    "__KEY": 15,  
    "__STAMP": 0,  
    "ID": 15,  
    "name": "Alpha North Yellow",  
    "creationDate": "!!0000-00-00!!",  
    "revenues": 82000000,  
    "extra": null,  
    "comments": "",  
    "__GlobalStamp": 0  
  },  
  {  
    "__KEY": 34,  
    "__STAMP": 0,  
    "ID": 34,  
    "name": "Astral Partner November",  
    "creationDate": "!!0000-00-00!!",  
    "revenues": 90000000,  
    "extra": null,  
    "comments": "",  
    "__GlobalStamp": 0  
  },  
  {  
    "__KEY": 47,  
    "__STAMP": 0,  
    "ID": 47,  
    "name": "Audio Production Uniform",  
    "creationDate": "!!0000-00-00!!",  
    "revenues": 28000000,  
    "extra": null,  
    "comments": "",  
    "__GlobalStamp": 0  
  }  
]
```

Los mismos datos en su formato JSON por defecto:

```
{
  "__entityModel": "Company",
  "__GlobalStamp": 50,
  "__COUNT": 52,
  "__FIRST": 0,
  "__ENTITIES": [
    {
      "__KEY": "15",
      "__TIMESTAMP": "2018-03-28T14:38:07.434Z",
      "__STAMP": 0,
      "ID": 15,
      "name": "Alpha North Yellow",
      "creationDate": "0!0!0",
      "revenues": 82000000,
      "extra": null,
      "comments": "",
      "__GlobalStamp": 0,
      "employees": {
        "__deferred": {
          "uri": "/rest/Company(15)/employees?$expand=employees"
        }
      }
    },
    {
      "__KEY": "34",
      "__TIMESTAMP": "2018-03-28T14:38:07.439Z",
      "__STAMP": 0,
      "ID": 34,
      "name": "Astral Partner November",
      "creationDate": "0!0!0",
      "revenues": 90000000,
      "extra": null,
      "comments": "",
      "__GlobalStamp": 0,
      "employees": {
        "__deferred": {
          "uri": "/rest/Company(34)/employees?$expand=employees"
        }
      }
    },
    {
      "__KEY": "47",
      "__TIMESTAMP": "2018-03-28T14:38:07.443Z",
      "__STAMP": 0,
      "ID": 47,
      "name": "Audio Production Uniform",
      "creationDate": "0!0!0",
      "revenues": 28000000,
      "extra": null,
      "comments": "",
      "__GlobalStamp": 0,
      "employees": {
        "__deferred": {
          "uri": "/rest/Company(47)/employees?$expand=employees"
        }
      }
    }
  ],
  "__SENT": 3
}
```

# \$atomic/\$atonce

Permite que las acciones de la solicitud REST estén en una transacción. Si no hay errores, la transacción se valida. En caso contrario, la transacción se cancela.

## Descripción

Cuando tiene varias acciones juntas, puede utilizar `$atomic/$atonce` para asegurarse de que ninguna de las acciones se complete si una de ellas falla. Puede utilizar `$atomic` o `$atonce`.

## Ejemplo

Llamamos a la siguiente petición REST en una transacción.

```
POST /rest/Employee?method=update&$atomic=true
```

Datos POST:

```
[  
 {  
   "__KEY": "200",  
   "firstname": "John"  
 },  
 {  
   "__KEY": "201",  
   "firstname": "Harry"  
 }  
 ]
```

Obtenemos el siguiente error en la segunda entidad y por lo tanto la primera entidad tampoco se guarda:

```
{
    "__STATUS": {
        "success": true
    },
    "__KEY": "200",
    "__STAMP": 1,
    "uri": "/rest/Employee(200)",
    "__TIMESTAMP": "!!2020-04-03!!",
    "ID": 200,
    "firstname": "John",
    "lastname": "Keeling",
    "isWoman": false,
    "numberOfKids": 2,
    "addressID": 200,
    "gender": false,
    "address": {
        "__deferred": {
            "uri": "/rest/Address(200)",
            "__KEY": "200"
        }
    },
    "__ERROR": [
        {
            "message": "Cannot find entity with \"201\" key in the \"Employee\" dataclass",
            "componentSignature": "dbmg",
            "errCode": 1542
        }
    ]
}
```

Aunque el salario de la primera entidad tiene un valor de 45000, este valor no se guardó en el servidor y tampoco se modificó el *timestamp* (*\_\_STAMP*). Si recargamos la entidad, veremos el valor anterior.

# \$attributes

Permite seleccionar los atributos relacionales a obtener de la dataclass (*por ejemplo*, `Company(1)?$attributes=employees.lastname` o `Employee?$attributes=employer.name`).

## Descripción

Cuando tenga atributos relacionales en una dataclass, utilice `$attributes` para definir la ruta de los atributos cuyos valores desea obtener para la entidad o entidades relacionadas.

Puede aplicar `$attributes` a una entidad (*p. Ej.*, `People(1)`) o una entity selection (*p. Ej.*, `People/$entityset/0AF4679A5C394746BFEB68D2162A19FF`).

- Si `$attributes` no se especifica en una consulta, o si se pasa el valor "\*", se extraen todos los atributos disponibles. Los atributos \*\*de entidad relacionada\*\* se extraen con la forma simple: un objeto con la propiedad `__KEY` (llave primaria) y `URI`. Los atributos de las entidades relacionadas no se extraen.
- Si se especifica `$attributes` para los atributos de entidad relacionada:
  - `$attributes=relatedEntity` : se devuelve la entidad relacionada con forma simple (propiedad `__KEY` diferida (llave primaria)) y `URI`.
  - `$attributes=relatedEntity.*` : se devuelven todos los atributos de la entidad relacionada
  - `$attributes=relatedEntity.attributePath1, relatedEntity.attributePath2, ...` : sólo se devuelven los atributos de la entidad relacionada.
- Si se especifica `$attributes` para los atributos de entidades relacionadas:
  - `$attributes=relatedEntities.*` : se devuelven todas las propiedades de todas las entidades relacionadas
  - `$attributes=relatedEntities.attributePath1, relatedEntities.attributePath2, ...` : sólo se devuelven los atributos de las entidades relacionadas.

## Ejemplo con varias entidades relacionadas

Si pasamos la petición REST siguiente para nuestra clase de datos Company (que tiene un atributo de relación "empleados"):

```
GET /rest/Company(1)/?$attributes=employees.lastname
```

Respuesta:

```
{
    "__entityModel": "Company",
    "__KEY": "1",
    "__TIMESTAMP": "2018-04-25T14:41:16.237Z",
    "__STAMP": 2,
    "employees": {
        "__ENTITYSET": "/rest/Company(1)/employees?$expand=employees",
        "__GlobalStamp": 50,
        "__COUNT": 135,
        "__FIRST": 0,
        "__ENTITIES": [
            {
                "__KEY": "1",
                "__TIMESTAMP": "2019-12-01T20:18:26.046Z",
                "__STAMP": 5,
                "lastname": "ESSEAL"
            },
            {
                "__KEY": "2",
                "__TIMESTAMP": "2019-12-04T10:58:42.542Z",
                "__STAMP": 6,
                "lastname": "JONES"
            },
            ...
        ]
    }
}
```

Si desea obtener todos los atributos de los empleados:

```
GET /rest/Company(1)/?$attributes=employees.*
```

Si quiere obtener el apellido y los atributos de nombre del trabajo de los empleados:

```
GET /rest/Company(1)/?$attributes=employees.lastname,employees.jobname
```

## Ejemplo con una entidad relacionada

Si pasamos la petición REST siguiente para nuestra clase de datos Employee (que tiene varios atributos relacionales, incluyendo "employer"):

```
GET /rest/Employee(1)?$attributes=employer.name
```

Respuesta:

```
{
    "__entityModel": "Employee",
    "__KEY": "1",
    "__TIMESTAMP": "2019-12-01T20:18:26.046Z",
    "__STAMP": 5,
    "employer": {
        "__KEY": "1",
        "__TIMESTAMP": "2018-04-25T14:41:16.237Z",
        "__STAMP": 0,
        "name": "Adobe"
    }
}
```

Si desea obtener todos los atributos del empleador:

```
GET /rest/Employee(1)?$attributes=employer.*
```

Si desea obtener los apellidos de todos los empleados de la empresa:

```
GET /rest/Employee(1)?$attributes=employer.employees.lastname
```

# \$binary

Pase "true" para guardar el BLOB como documento (también debe pasar `$expand={blobAttributeName}`)

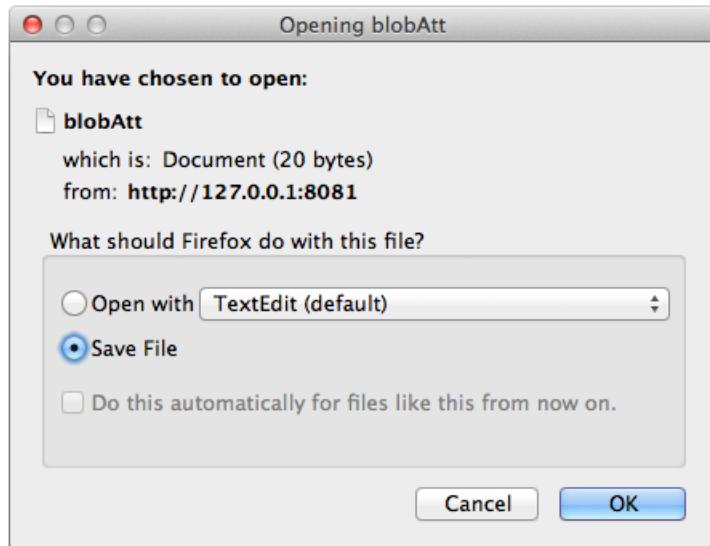
## Descripción

`$binary` permite guardar el BLOB como un documento. También debe utilizar el comando `$expand` junto con él.

Cuando haga la siguiente petición:

```
GET /rest/Company(11)/blobAtt?$binary=true&$expand=blobAtt
```

Se le preguntará dónde guardar el BLOB en el disco:



# \$compute

Cálculo de atributos específicos (e., `Employee/salary/?$compute=sum`) o en el caso de un atributo Objeto (*por ejemplo*, `Employee/objectAtt.property1/?$compute=sum`)

## Descripción

Este parámetro le permite realizar cálculos sobre sus datos.

Si desea realizar un cálculo sobre un atributo, escriba lo siguiente:

```
GET /rest/Employee/salary/?$compute=$all
```

Si desea pasar un atributo Objeto, debe pasar una de sus propiedades. Por ejemplo:

```
GET /rest/Employee/objectAtt.property1/?$compute=$all
```

Puede utilizar cualquiera de las siguientes palabras claves:

Palabras clave	Descripción
\$all	Un objeto JSON que define todas las funciones del atributo (promedio, recuento, mínimo, máximo y suma para los atributos de tipo Number y count, mínimo y máximo para los atributos de tipo String)
average	Obtener la media de un atributo numérico
count	Obtener el número total en la colección o en la clase de datos (en ambos casos hay que especificar un atributo)
min	Obtener el valor mínimo de un atributo numérico o el valor más bajo en un atributo de tipo String
max	Obtener el valor máximo de un atributo numérico o el valor más alto en un atributo de tipo String
sum	Obtener la suma de un atributo numérico

## Ejemplo

Si desea obtener todos los cálculos de un atributo de tipo Número, puede escribir:

```
GET /rest/Employee/salary/?$compute=$all
```

Respuesta:

```
{
  "salary": {
    "count": 4,
    "sum": 335000,
    "average": 83750,
    "min": 70000,
    "max": 99000
  }
}
```

Si desea obtener todos los cálculos de un atributo de tipo String, puede escribir:

```
GET /rest/Employee/firstName/?$compute=$all
```

Respuesta:

```
{  
    "salary": {  
        "count": 4,  
        "min": Anne,  
        "max": Victor  
    }  
}
```

Si desea obtener un cálculo con un atributo, escriba lo siguiente:

```
GET /rest/Employee/salary/?$compute=sum
```

Respuesta:

```
235000
```

Si desea realizar un cálculo con un atributo Objeto, escriba lo siguiente:

```
GET /rest/Employee/objectAttribute.property1/?$compute=sum
```

Respuesta:

```
45
```

# \$distinct

Devuelve los diferentes valores de un atributo específico en una colección ( *por ejemplo*, `Company/name? $filter="name=a*&$distinct=true` )

## Descripción

`$distinct` permite devolver una colección que contiene los diferentes valores de una petición sobre un atributo específico. Sólo se puede especificar un atributo en la dataclass. Generalmente, el tipo String es el mejor; sin embargo, también puede utilizarlo en cualquier tipo de atributo que pueda contener múltiples valores.

También puede utilizar `$skip` y `$top/$limit`, si desea navegar por la selección antes de colocarla en un array.

## Ejemplo

En nuestro ejemplo siguiente, queremos recuperar los diferentes valores de un nombre de empresa que empiece por la letra "a":

```
GET /rest/Company/name?$filter="name=a*&$distinct=true
```

Respuesta:

```
[  
    "Adobe",  
    "Apple"  
]
```

# \$entityset

Después de [crear un conjunto de entidades](#) mediante el uso de `$method=entityset`, puede utilizarlo posteriormente.

## Sintaxis disponible

Sintaxis	Ejemplo	Descripción
<code>\$entityset/{entitySetID}</code>	<code>/People/\$entityset/0ANUMBER</code>	Recupera un conjunto de entidades existente
<code>\$entityset/{entitySetID}?\$operator...&amp;\$otherCollection</code>	<code>/Employee/\$entityset/0ANUMBER? \$logicOperator=AND &amp;\$otherCollection=C0ANUMBER</code>	Crea un nuevo conjunto de entidades a partir de la comparación de conjuntos de entidades existentes

## \$entityset/{entitySetID}

Recupera un conjunto de entidades existente (p. ej., `People/$entityset/0AF4679A5C394746BFEB68D2162A19FF`)

### Descripción

Esta sintaxis permite ejecutar cualquier operación sobre un conjunto de entidades definido.

Como los conjuntos de entidades tienen un límite de tiempo (ya sea por defecto o después de llamar a `$timeout` con su propio límite), puede llamar a `$savedfilter` y a `$savedorderby` para guardar el filtro y ordenar por instrucciones cuando cree un conjunto de entidades.

Cuando se recupera un conjunto de entidades existente almacenado en la caché de 4D Server, también se puede aplicar cualquiera de los siguientes elementos al conjunto de entidades: `$expand`, `$filter`, `$orderby`, `$skip`, y `$top/$limit`.

### Ejemplo

Después de crear un conjunto de entidades, el ID del conjunto de entidades se devuelve junto con los datos. Llame a este ID de la siguiente manera:

```
GET /rest/Employee/$entityset/9718A30BF61343C796345F3BE5B01CE7
```

## \$entityset/{entitySetID}?\$operator...&\$otherCollection

Cree otro conjunto de entidades basado en conjuntos de entidades creados anteriormente

Parámetros	Tipo	Descripción
<code>\$operator</code>	Cadena	Uno de los operadores lógicos para probar con el otro conjunto de entidades
<code>\$otherCollection</code>	Cadena	ID del conjunto de entidades

### Descripción

Después de crear un conjunto de entidades (conjunto de entidades nº 1) utilizando `$method=entityset`, puede crear otro conjunto de entidades utilizando la sintaxis `$entityset/{entitySetID}?$operator... &$otherCollection`, la propiedad `$operator` (cuyos valores se muestran a continuación), y otro conjunto de entidades (conjunto de entidades nº 2) definido por la propiedad `$otherCollection`. Los dos conjuntos de entidades deben estar en la misma clase de datos.

A continuación, puede crear otro conjunto de entidades que contenga los resultados de esta llamada utilizando el `$method=entityset` al final de la petición REST.

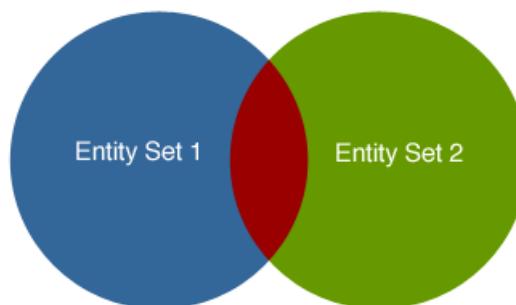
Aquí están los operadores lógicos:

Operador	Descripción
AND	Devuelve las entidades comunes a ambos conjuntos de entidades
O	Devuelve las entidades contenidas en ambos conjuntos de entidades
EXCEPT	Devuelve las entidades del conjunto de entidades #1 menos las del conjunto de entidades #2
INTERSECT	Devuelve true o false si hay una intersección de las entidades en ambos conjuntos de entidades (lo que significa que al menos una entidad es común en ambos conjuntos de entidades)

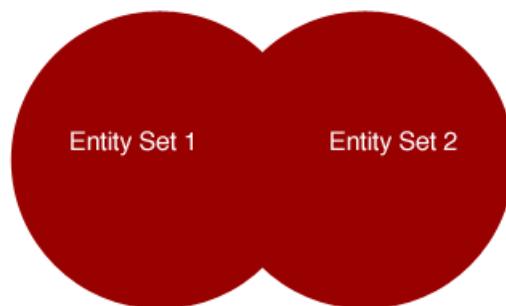
Los operadores lógicos no distinguen entre mayúsculas y minúsculas, por lo que puede escribir "AND" o "and".

A continuación se muestra una representación de los operadores lógicos basada en dos conjuntos de entidades. La sección roja es la que se devuelve.

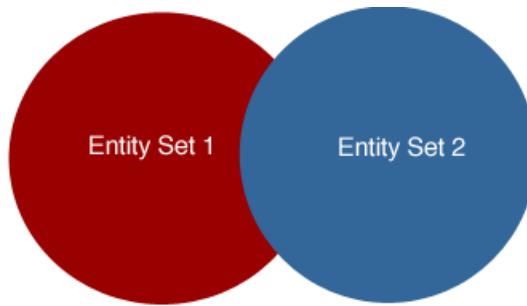
AND



O



EXCEPT



La sintaxis es la siguiente:

```
GET /rest/dataClass/$entityset/entitySetID?$logicOperator=AND&$otherCollection=entitySetID
```

## Ejemplo

En el ejemplo siguiente, devolvemos las entidades que están en ambos conjuntos de entidades ya que estamos utilizando el operador lógico AND:

```
GET /rest/Employee/$entityset/9718A30BF61343C796345F3BE5B01CE7?  
$logicOperator=AND&$otherCollection=C05A0D887C664D4DA1B38366DD21629B
```

Si queremos saber si los dos conjuntos de entidades se cruzan, podemos escribir lo siguiente:

```
GET /rest/Employee/$entityset/9718A30BF61343C796345F3BE5B01CE7?  
$logicOperator=intersect&$otherCollection=C05A0D887C664D4DA1B38366DD21629B
```

Si hay una intersección, esta consulta devuelve true. En caso contrario, devuelve false.

En el siguiente ejemplo creamos un nuevo conjunto de entidades que combina todas las entidades de ambos conjuntos de entidades:

```
GET /rest/Employee/$entityset/9718A30BF61343C796345F3BE5B01CE7?  
$logicOperator=OR&$otherCollection=C05A0D887C664D4DA1B38366DD21629B&$method=entityset
```

## \$expand

Expande una imagen almacenada en un atributo Image ( *p. ej.*, Employee(1)/photo?

```
$imageformat=best&$expand=photo )
```

o

Expande un atributo BLOB para guardarla.

Compatibilidad: por razones de compatibilidad, \$expand puede utilizarse para expandir un atributo relacional (*por ejemplo*, Company(1)?\$expand=staff o Employee/?\$filter="firstName BEGIN a" & \$expand=employer ). Sin embargo, se recomienda utilizar `$attributes` para esta funcionalidad.

## Visualización de un atributo de imagen

Si desea visualizar un atributo de imagen en su totalidad, escriba lo siguiente:

```
GET /rest/Employee(1)/photo?$imageformat=best&$version=1&$expand=photo
```

Para más información sobre los formatos de imagen, consulte `$imageformat`. Para más información sobre el parámetro de versión, consulte `$version`.

## Guardar un atributo BLOB en el disco

Si quiere guardar un BLOB almacenado en su clase de datos, puedes escribir lo siguiente pasando también "true" a `$binary`:

```
GET /rest/Company(11)/blobAtt?$binary=true&$expand=blobAtt
```

# \$filter

Permite consultar los datos de una clase de datos o de un método (*p. ej., \$filter="firstName!= '' AND salary>30000"* )

## Descripción

Este parámetro le permite definir el filtro para su clase de datos o método.

### Utilizar un filtro simple

Un filtro se compone de los siguientes elementos:

{attribute} {comparator} {value}

Por ejemplo: `$filter="firstName=john"` donde `firstName` es el atributo, `=` es el comparador y `john` es el valor.

### Utilizar un filtro complejo

Un filtro más complejo se compone de los siguientes elementos, que unen dos consultas:

{attribute} {comparator} {value} {AND/OR/EXCEPT} {attribute} {comparator} {value}

Por ejemplo: `$filter="firstName=john AND salary>20000"` donde `firstName` y `salary` son atributos de la clase de datos Employee.

### Utilizar la propiedad params

También puede utilizar la propiedad params de 4D.

{attribute} {comparator} {placeholder} {AND/OR/EXCEPT} {attribute} {comparator} {placeholder}&\$params='["{value1}","{value2}"]'"

Por ejemplo: `$filter="firstName=:1 AND salary:>2"&$params='["john",20000]'`  donde `firstName` y `salary` son los atributos de la clase de datos Employee.

Para más información sobre cómo consultar los datos en 4D, consulte la [dataClass.query\(\)](#) documentación.

Al insertar comillas ('') o comillas dobles (""), debe escaparlas utilizando su código de caracteres:

- Comillas (''): \u0027
- Comillas dobles (""): \u0022

Por ejemplo, se puede escribir lo siguiente al pasar un valor con una comilla cuando se utiliza la propiedad `params`:

`http://127.0.0.1:8081/rest/Person/?$filter="lastName=:1"&$params='["\u0027Reilly"]'`

Si pasa el valor directamente, puede escribir lo siguiente: `http://127.0.0.1:8081/rest/Person/?$filter="lastName=0'Reilly"`

## Atributo

Si el atributo está en la misma clase de datos, puede pasarlo directamente (*p. ej., firstName*). Sin embargo, si quiere consultar otra clase de datos, debe incluir el nombre del atributo relacional y el nombre del atributo, es decir, la ruta de acceso (*por ejemplo, nombre.empleador*). El nombre del atributo distingue entre mayúsculas y minúsculas (`firstName` no es igual a `FirstName` ).

También puede consultar los atributos de tipo Object utilizando la anotación de puntos. Por ejemplo, si tiene un atributo cuyo nombre es "objAttributo" con la siguiente estructura:

```
{  
    prop1: "this is my first property",  
    prop2: 9181,  
    prop3: ["abc", "def", "ghi"]  
}
```

Puede buscar en el objeto escribiendo lo siguiente:

```
GET /rest/Person/?filter="objAttribute.prop2 == 9181"
```

## Comparador

El comparador debe ser uno de los siguientes valores:

Comparador	Descripción
=	igual a
!=	diferente de
>	mayor que
>=	mayor o igual que
<	menor que
<=	menor o igual que
begin	comienza con

## Ejemplos

En el siguiente ejemplo, buscamos a todos los empleados cuyo apellido empieza por "j":

```
GET /rest/Employee?$filter="lastName begin j"
```

En este ejemplo, buscamos en la clase de datos Empleado todos los empleados cuyo salario sea superior a 20.000 y que no trabajen para una empresa llamada Acme:

```
GET /rest/Employee?$filter="salary>20000 AND  
employer.name!=acme"&$orderby="lastName,firstName"
```

En este ejemplo, buscamos en la clase de datos Person todas las personas cuya propiedad número en el atributo anotherobj de tipo Object es mayor que 50:

```
GET /rest/Person/?filter="anotherobj.myNum > 50"
```

# \$imageformat

Define qué formato de imagen utilizar para recuperar imágenes (por ejemplo, `$imageformat=png` )

## Descripción

Definir el formato a utilizar para mostrar las imágenes. Por defecto, se elegirá el mejor formato para la imagen. No obstante, puede seleccionar uno de los siguientes formatos:

Tipo	Descripción
GIF	Formato GIF
PNG	Formato PNG
JPEG	Formato JPEG
TIFF	Formato TIFF
best	El mejor formato en función de la imagen

Una vez que haya definido el formato, debe pasar el atributo de imagen a `$expand` para cargar la foto completamente.

Si no hay ninguna imagen para cargar o el formato no permite cargar la imagen, la respuesta estará vacía.

## Ejemplo

El siguiente ejemplo define el formato de la imagen a JPEG independientemente del tipo real de la foto y pasa el número de versión real enviado por el servidor:

```
GET /rest/Employee(1)/photo?$imageformat=jpeg&$version=3&$expand=photo
```

# \$lock

Locks and unlocks an entity using the [pessimistic mechanism](#).

## Sintaxis

To lock an entity for other sessions and 4D processes:

```
/?$lock=true
```

To unlock the entity for other sessions and 4D processes:

```
/?$lock=false
```

The `lockKindText` property is "Locked by session".

## Descripción

The locks triggered by the REST API are put at the [session](#) level.

A locked entity is seen as *locked* (i.e. lock / unlock / update / delete actions are not possible) by:

- otras sesiones REST
- 4D processes (client/server, remote datastore, standalone) running on the REST server.

An entity locked by the REST API can only be unlocked:

- by its locker, i.e. a `/?$lock=false` in the REST session that sets `/?$lock=true`
- or if the session's [inactivity timeout](#) is reached (the session is closed).

## Respuesta

A `?$lock` request returns a JSON object with `"result=true` if the lock operation was successful and `"result=false` if it failed.

The returned "`__STATUS`" object has the following properties:

Propiedad		Tipo	Descripción
			<i>Available only in case of success:</i>
success		booleano	true if the lock action is successful (or if the entity is already locked in the current session), false otherwise (not returned in this case).
			<i>Disponible sólo en caso de error:</i>
status		number	Código de error, ver abajo
statusText		texto	Descripción del error, ver abajo
lockKind		number	Código de bloqueo
lockKindText		texto	"Locked by session" if locked by a REST session, "Locked by record" if locked by a 4D process
lockInfo		objeto	Information about the lock origin. Returned properties depend on the lock origin (4D process or REST session).
			<i>Disponible sólo para un bloqueo por proceso 4D:</i>
	task_id	number	ID del Proceso
	user_name	texto	Nombre de usuario de la sesión en la máquina
	user4d_alias	texto	Nombre o alias del usuario 4D
	user4d_id	number	ID del usuario en el directorio de la base de datos 4D
	host_name	texto	Nombre de la máquina
	task_name	texto	Nombre del proceso
	client_version	texto	Versión del cliente
			<i>Disponible sólo para un bloqueo por sesión REST:</i>
	host	texto	URL que bloqueó la entidad (por ejemplo, "127.0.0.1:8043")
	IPAddr	texto	Dirección IP del bloqueo (por ejemplo: "127.0.0.1")
	recordNumber	number	Record number of the locked record
	userAgent	texto	userAgent of the locker (e.g. Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36")

The following values can be returned in the `status` and `statusText` properties of the `__STATUS` object in case of error:

status	statusText	Comentario
2	"Stamp has changed"	El valor del sello interno de la entidad no coincide con el de la entidad almacenada en los datos (bloqueo optimista).
3	"Already locked"	La entidad está bloqueada por un bloqueo pesimista.
4	"Other error"	Un error grave es un error de base de datos de bajo nivel (por ejemplo, una llave duplicada), un error de hardware, etc.
5	"Entity does not exist anymore"	La entidad ya no existe en los datos.

## Ejemplo

Bloqueamos una entidad en un primer navegador:

```
GET /rest/Customers(1)/?$lock=true
```

Respuesta:

```
{  
    "result": true,  
    "__STATUS": {  
        "success": true  
    }  
}
```

In a second browser (other session), we send the same request.

Respuesta:

```
{  
    "result":false,  
    "__STATUS":{  
        "status":3,  
        "statusText":"Already Locked",  
        "lockKind":7,  
        "lockKindText":"Locked By Session",  
        "lockInfo":{  
            "host":"127.0.0.1:8043",  
            "IPAddr":"127.0.0.1",  
            "recordNumber": 7,  
            "userAgent": """Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_3) AppleWebKit/537.36..."  
        }  
    }  
}
```

# \$method

Este parámetro permite definir la operación a ejecutar con la entidad o selección de entidades devuelta.

## Sintaxis disponible

Sintaxis	Ejemplo	Descripción
<code>\$method=delete</code>	<code>POST /Employee?\$filter="ID=11"&amp;\$method=delete</code>	Elimina la entidad actual, la colección de entidades o la selección de entidades
<code>\$method=entityset</code>	<code>GET /People/?\$filter="ID&gt;320"&amp;\$method=entityset&amp; \$timeout=600</code>	Crea un conjunto de entidades en la caché de 4D Server basado en la colección de entidades definidas en la solicitud REST
<code>\$method=release</code>	<code>GET /Employee/\$entityset/&lt;entitySetID&gt;?&amp;\$method=release</code>	Libera un conjunto de entidades existente almacenado en la caché de 4D Server
<code>\$method=subentityset</code>	<code>GET /Company(1)/staff?&amp;\$expand=staff&amp;&amp;\$method=subentityset&amp;&amp;\$subOrderby=lastName ASC</code>	Crea un conjunto de entidades basado en la colección de entidades relativas definidas en la petición REST
<code>\$method=update</code>	<code>POST /Person/?\$method=update</code>	Actualiza y/o crea una o varias entidades

## \$method=delete

Elimina la entidad actual, la colección de entidades o la selección de entidad actual (creada vía REST)

### Descripción

Con `$method=delete`, puede eliminar una entidad o una colección de entidades entera. Puede definir la colección de entidades utilizando, por ejemplo, `$filter` o especificando uno directamente mediante `{dataClass}({key})` (por ejemplo, `/Employee(22)`).

También puede eliminar las entidades de un conjunto de entidades, llamando a `$entityset/{entitySetID}`.

## Ejemplo

A continuación, puede escribir la siguiente petición REST para eliminar la entidad cuya llave es 22:

```
POST /rest/Employee(22)/?$method=delete
```

También se puede hacer una petición de información utilizando `$filter`:

```
POST /rest/Employee?$filter="ID=11"&$method=delete
```

También puede eliminar un conjunto de entidades utilizando `$entityset/{entitySetID}`:

```
POST /rest/Employee/$entityset/73F46BE3A0734EAA9A33CA8B14433570?&$method=delete
```

Respuesta:

```
{
  "ok": true
}
```

## \$method=entityset

Crea un conjunto de entidades en la caché de 4D Server basado en la colección de entidades definidas en la solicitud REST

### Descripción

Cuando se crea una colección de entidades en REST, también se puede crear un conjunto de entidades que se guardará en la caché de 4D Server. El conjunto de entidades tendrá un número de referencia que puede pasar a `$entityset/{entitySetID}` para acceder a él. Por defecto, es válido durante dos horas; sin embargo, puede modificar esa cantidad de tiempo pasando un valor (en segundos) a `$timeout`.

Si ha utilizado `$savedfilter` y/o `$savedorderby` (junto con `$filter` y/o `$orderby`) cuando creó su conjunto de entidades, puede volver a crearlo con el mismo ID de referencia aunque se haya eliminado de la caché de 4D Server.

### Ejemplo

Para crear un conjunto de entidades, que se guardará en la caché de 4D Server durante dos horas, añada `$method=entityset` al final de su petición REST:

```
GET /rest/People/?$filter="ID>320"&$method=entityset
```

Puede crear un conjunto de entidades que se almacenará en la caché de 4D Server durante sólo diez minutos pasando un nuevo tiempo de espera a `$timeout`:

```
GET /rest/People/?$filter="ID>320"&$method=entityset&$timeout=600
```

También puede guardar el filtro y ordenar por, pasando true a `$savedfilter` y `$savedorderby`.

`$skip` y `$top/$limit` no se tienen en cuenta al guardar un conjunto de entidades.

Después de crear un conjunto de entidades, el primer elemento, `__ENTITYSET`, se añade al objeto devuelto e indica la URI a utilizar para acceder al conjunto de entidades:

```
__ENTITYSET: "http://127.0.0.1:8081/rest/Employee/$entityset/9718A30BF61343C796345F3BE5B01CE7"
```

## \$method=release

Libera un conjunto de entidades existente almacenado en la caché de 4D Server.

### Descripción

Puede liberar un conjunto de entidades, que creó utilizando `$method=entityset`, de la caché de 4D Server.

### Ejemplo

Muestra un conjunto de entidades existente:

```
GET /rest/Employee/$entityset/4C51204DD8184B65AC7D79F09A077F24?method=release
```

Respuesta:

Si la petición fue exitosa, se devuelve la siguiente respuesta:

```
{
  "ok": true
}
Si no se encuentra el conjunto de entidades, se devuelve un error:

{
  "__ERROR": [
    {
      "message": "Error code: 1802\nEntitySet \\"4C51204DD8184B65AC7D79F09A077F24\\" cannot be found",
      "componentSignature": "dbmg",
      "errCode": 1802
    }
  ]
}
```

## \$method=subentityset

Crea un conjunto de entidades en la caché de 4D Server basado en la colección de entidades relativas definidas en la petición REST

### Descripción

`$method=subentityset` permite ordenar los datos devueltos por el atributo relacional definido en la petición REST.

Para ordenar los datos, se utiliza la propiedad `$sub0rderby`. Para cada atributo, se define el orden como ASC (o asc) para el orden ascendente y DESC (desc) para el orden descendente. Por defecto, los datos se clasifican en orden ascendente.

Si desea especificar varios atributos, puede delimitarlos con una coma, µ, `$sub0rderby="lastName desc, firstName asc"`.

### Ejemplo

Si quiere recuperar sólo las entidades relacionadas para una entidad específica, puede hacer la siguiente petición REST donde personal es el atributo de relación en la clase de datos Company vinculada a la clase de datos Employee:

```
GET /rest/Company(1)/staff?$expand=staff&$method=subentityset&$sub0rderby=lastName ASC
```

Respuesta:

```
{
  "__ENTITYSET": "/rest/Employee/$entityset/FF625844008E430B9862E5FD41C741AB",
  "__entityModel": "Employee",
  "__COUNT": 2,
  "__SENT": 2,
  "__FIRST": 0,
  "__ENTITIES": [
    {
      "__KEY": "4",
      "__STAMP": 1,
      "ID": 4,
      "firstName": "Linda",
      "lastName": "Jones",
      "birthday": "1970-10-05T14:23:00Z",
      "employer": {
        "__deferred": {
          "uri": "/rest/Company(1)",
          "__KEY": "1"
        }
      }
    },
    {
      "__KEY": "1",
      "__STAMP": 3,
      "ID": 1,
      "firstName": "John",
      "lastName": "Smith",
      "birthday": "1985-11-01T15:23:00Z",
      "employer": {
        "__deferred": {
          "uri": "/rest/Company(1)",
          "__KEY": "1"
        }
      }
    }
  ]
}
```

## \$method=update

Actualiza y/o crea una o varias entidades

### Descripción

`$method=update` le permite actualizar y/o crear una o más entidades en un solo POST. Si se actualiza y/o crea una entidad, se efectúa en un objeto con, para cada propiedad, un atributo y su valor, *por ejemplo* `{ lastName: "Smith" }`. Si actualiza y/o crea varias entidades, debe crear una colección de objetos.

En cualquier caso, debe definir los POST datos en el body de la petición.

Para actualizar una entidad, debes pasar los parámetros `__KEY` y `__STAMP` en el objeto junto con todos los atributos modificados. Si faltan ambos parámetros, se añadirá una entidad con los valores del objeto que envíe en el cuerpo de su POST.

Los triggers se ejecutan inmediatamente al guardar la entidad en el servidor. La respuesta contiene todos los datos tal y como existen en el servidor.

También puede poner estas solicitudes para crear o actualizar entidades en una transacción llamando a `$atomic/$atonce`. Si se produce algún error durante la validación de los datos, no se guarda ninguna de las entidades. También puede utilizar `$method=validate` para validar las entidades antes de crearlas o actualizarlas.

Si surge un problema al añadir o modificar una entidad, se le devolverá un error con esa información.

#### Notas para tipos de atributos específicos:

- Las fechas deben expresarse en formato JS: YYYY-MM-DDTHH:MM:SSZ (por ejemplo, "2010-10-05T23:00:00Z"). Si ha seleccionado la propiedad Fecha únicamente para su atributo Fecha, se eliminará la zona horaria y la hora (hora, minutos y segundos). En este caso, también puede enviar la fecha en el formato que se le devuelve dd!mm!aaaa (por ejemplo, 05!10!2013).
- Booleanos son true o false.
- Los archivos subidos mediante `$upload` pueden aplicarse a un atributo de tipo Imagen o BLOB pasando el objeto devuelto en el siguiente formato `{"ID": "D507BC03E613487E9B4C2F6A0512FE50"}`

## Ejemplo

Para actualizar una entidad específica, se utiliza la siguiente URL:

```
POST /rest/Person/?$method=update
```

Datos POST:

```
{
  __KEY: "340",
  __STAMP: 2,
  firstName: "Pete",
  lastName: "Miller"
}
```

Los atributos `firstName` y `lastName` de la entidad indicada anteriormente se modificarán dejando todos los demás atributos (excepto los calculados basados en estos atributos) sin cambios.

Si quiere crear una entidad, puede enviar, vía POST los atributos utilizando esta URL:

```
POST /rest/Person/?$method=update
```

Datos POST:

```
{
  firstName: "John",
  lastName: "Smith"
}
```

También puede crear y actualizar múltiples entidades al mismo tiempo utilizando la misma URL anterior pasando múltiples objetos en un array al POST:

```
POST /rest/Person/?$method=update
```

Datos POST:

```
[{
  "__KEY": "309",
  "__STAMP": 5,
  "ID": "309",
  "firstName": "Penelope",
  "lastName": "Miller"
}, {
  "firstName": "Ann",
  "lastName": "Jones"
}]
```

Respuesta:

Cuando añade o modifica una entidad, se devuelve con los atributos modificados. Por ejemplo, si se crea el nuevo empleado anterior, se devolverá lo siguiente:

```
{  
    "__KEY": "622",  
    "__STAMP": 1,  
    "uri": "http://127.0.0.1:8081/rest/Employee(622)",  
    "__TIMESTAMP": "!!2020-04-03!!",  
    "ID": 622,  
    "firstName": "John",  
    "firstName": "Smith"  
}
```

Si, por ejemplo, el sello no es correcto, se devuelve el siguiente error:

```
{
    "__STATUS": {
        "status": 2,
        "statusText": "Stamp has changed",
        "success": false
    },
    "__KEY": "1",
    "__STAMP": 12,
    "__TIMESTAMP": "!!2020-03-31!!",
    "ID": 1,
    "firstname": "Denise",
    "lastname": "O'Peters",
    "isWoman": true,
    "numberOfKids": 1,
    "addressID": 1,
    "gender": true,
    "imageAtt": {
        "__deferred": {
            "uri": "/rest/Persons(1)/imageAtt?$imageformat=best&$version=12&$expand=imageAtt",
            "image": true
        }
    },
    "extra": {
        "num": 1,
        "alpha": "I am 1"
    },
    "address": {
        "__deferred": {
            "uri": "/rest/Address(1)",
            "__KEY": "1"
        }
    },
    "__ERROR": [
        {
            "message": "Given stamp does not match current one for record# 0 of table Persons",
            "componentSignature": "dbmg",
            "errCode": 1263
        },
        {
            "message": "Cannot save record 0 in table Persons of database remote_dataStore",
            "componentSignature": "dbmg",
            "errCode": 1046
        },
        {
            "message": "The entity# 1 in the \"Persons\" dataclass cannot be saved",
            "componentSignature": "dbmg",
            "errCode": 1517
        }
    ]
}{}}
}
```

# \$orderby

Ordena los datos devueltos por el atributo y el orden de clasificación definidos (*por ejemplo*, `$orderby="lastName desc, salary asc"`)

## Descripción

`$orderby` ordena las entidades devueltas por la petición REST. Para cada atributo, se especifica el orden como `ASC` (o `asc`) para el orden ascendente y `DESC` ( `desc` ) para el orden descendente. Por defecto, los datos se clasifican en orden ascendente. Por defecto, los datos se clasifican en orden ascendente.

## Ejemplo

En este ejemplo, recuperamos las entidades y las ordenamos al mismo tiempo:

```
GET /rest/Employee/?$filter="salary!=0"&$orderby="salary DESC, lastName ASC, firstName ASC"
```

El ejemplo siguiente ordena el conjunto de entidades por el atributo `lastName` en orden ascendente:

```
GET /rest/Employee/$entityset/CB1BCC603DB0416D939B4ED379277F02?$orderby="lastName"
```

Resultado:

```
{
  __entityModel: "Employee",
  __COUNT: 10,
  __SENT: 10,
  __FIRST: 0,
  __ENTITIES: [
    {
      __KEY: "1",
      __STAMP: 1,
      firstName: "John",
      lastName: "Smith",
      salary: 90000
    },
    {
      __KEY: "2",
      __STAMP: 2,
      firstName: "Susan",
      lastName: "O'Leary",
      salary: 80000
    },
    // more entities
  ]
}
```

# \$querypath

Devuelve la petición tal y como se ejecutó por 4D Server (\*por ejemplo, \*, \$querypath=true )

## Descripción

\$querypath devuelve la petición tal y como fue ejecutada por 4D Server. Si, por ejemplo, una parte de la petición pasada no devuelve ninguna entidad, el resto de la petición no se ejecuta. La petición de consulta está optimizada como se puede ver en este \$querypath .

Para más información sobre las rutas de petición, consulte [queryPlan y queryPath](#).

En la colección de pasos, hay un objeto con las siguientes propiedades que definen la petición ejecutada:

Propiedad	Tipo	Descripción
description	Cadena	Petición ejecutada o "AND" cuando hay varios pasos
time	Número	Número de milisegundos necesarios para ejecutar la petición
recordsfounds	Número	Número de registros encontrados
steps	Collection	Una colección con un objeto que define el siguiente paso de la ruta de la petición

## Ejemplo

Si pasó la siguiente petición:

```
GET /rest/Employee/$filter="employer.name=acme AND lastName=Jones"&$querypath=true
```

Y no se encontraron entidades, se devolvería la siguiente ruta de petición, si escribe lo siguiente:

```
GET /rest/$querypath
```

Respuesta:

```

__queryPath: {

    steps: [
        {
            description: "AND",
            time: 0,
            recordsfounds: 0,
            steps: [
                {
                    description: "Join on Table : Company : People.employer = Company. ID",
                    time: 0,
                    recordsfounds: 0,
                    steps: [
                        {
                            steps: [
                                {
                                    description: "Company.name = acme",
                                    time: 0,
                                    recordsfounds: 0
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    ]
}

```

Si, por el contrario, la primera consulta devuelve más de una entidad, se ejecutará la segunda. Si ejecutamos la siguiente consulta:

```
GET /rest/Employee/$filter="employer.name=a* AND lastName!=smith"&$querypath=true
```

Si se encuentra al menos una entidad, se devolverá la siguiente ruta de consulta, si se escribe lo siguiente:

```
GET /rest/$querypath
```

Respuesta:

```
"__queryPath": {
  "steps": [
    {
      "description": "AND",
      "time": 1,
      "recordsfounds": 4,
      "steps": [
        {
          "description": "Join on Table : Company : Employee.employer = Company. ID",
          "time": 1,
          "recordsfounds": 4,
          "steps": [
            {
              "steps": [
                {
                  "description": "Company.name LIKE a*",
                  "time": 0,
                  "recordsfounds": 2
                }
              ]
            }
          ]
        },
        {
          "description": "Employee.lastName # smith",
          "time": 0,
          "recordsfounds": 4
        }
      ]
    }
  ]
}
```

# \$queryplan

Devuelve la petición tal y como se pasó a 4D Server (\*por ejemplo, \*, \$queryplan=true )

## Descripción

\$queryplan devuelve el plan de la petición tal y como se pasó a 4D Server.

Propiedad	Tipo	Descripción
item	Cadena	Petición ejecutada
subquery	Array	Si hay una subconsulta, un objeto adicional que contiene una propiedad de elemento (como la anterior)

Para más información sobre los planes de petición, consulte [queryPlan](#) y [queryPath](#).

## Ejemplo

Si pasas la siguiente petición:

```
GET /rest/People/$filter="employer.name=acme AND lastName=Jones"&$queryplan=true
```

Respuesta:

```
__queryPlan: {
    And: [
        {
            item: "Join on Table : Company : People.employer = Company. ID",
            subquery: [
                {
                    item: "Company.name = acme"
                }
            ]
        },
        {
            item: "People.lastName = Jones"
        }
    ]
}
```

# \$savedfilter

Guarda el filtro definido por \$filter al crear un conjunto de entidades (*por ejemplo*, `$savedfilter="{filter}"`)

## Descripción

Cuando se crea un conjunto de entidades, se puede guardar el filtro que se ha utilizado para crearlo como medida de seguridad. Si el conjunto de entidades que ha creado es eliminado de la caché de 4D Server (debido al tiempo de espera, a la necesidad de espacio del servidor o a que lo ha eliminado llamando a `$method=release`).

Utilice `$savedfilter` para guardar el filtro que definió al crear su conjunto de entidades y luego pase `$savedfilter` junto con su llamada para recuperar cada vez el conjunto de entidades.

Si el conjunto de entidades ya no está en la caché de 4D Server, se recreará con un nuevo tiempo de espera de 10 minutos por defecto. El conjunto de entidades se refrescará (pueden incluirse ciertas entidades y eliminarse otras) desde la última vez que se creó, si ya no existía antes de recrearlo.

Si ha utilizado a la vez `$savedfilter` y `$savedorderby` en su llamada al crear un conjunto de entidades y luego omite uno de ellos, el nuevo conjunto de entidades, que tendrá el mismo número de referencia, lo reflejará.

## Ejemplo

En nuestro ejemplo, primero llamamos a ``\$savedfilter`` con la llamada inicial para crear un conjunto de entidades como se muestra a continuación:

```
GET /rest/People/?$filter="employer.name=Apple"&$savedfilter="employer.name=Apple"&$method=entityset
```

A continuación, cuando se accede al conjunto de entidades, se escribe lo siguiente para garantizar que el conjunto de entidades sea siempre válido:

```
GET /rest/People/$entityset/AEA452C2668B4F6E98B6FD2A1ED4A5A8?&$savedfilter="employer.name=Apple"
```

# \$savedorderby

Guarda el filtro definido por `$orderby` al crear un conjunto de entidades (*por ejemplo*, `$savedorderby="{orderby}"`)

## Descripción

Cuando se crea un conjunto de entidades, se puede guardar el sentido de la ordenación junto con el filtro utilizado para su creación como medida de seguridad. Si el conjunto de entidades que ha creado es eliminado de la caché de 4D Server (debido al tiempo de espera, a la necesidad de espacio del servidor o a que lo ha eliminado llamando a `$method=release`).

Utilice `$savedorderby` para guardar el orden que definió al crear su conjunto de entidades, luego pase `$savedorderby` junto con su llamada para recuperar cada vez el conjunto de entidades.

Si el conjunto de entidades ya no está en la caché de 4D Server, se recreará con un nuevo tiempo de espera de 10 minutos por defecto. Si ha utilizado tanto `$savedfilter` y `$savedorderby` en su llamada al crear un conjunto de entidades y luego omite uno de ellos, el nuevo conjunto de entidades, que tiene el mismo número de referencia, lo reflejará.

## Ejemplo

Primero se llama a `$savedorderby` con la llamada inicial para crear un conjunto de entidades:

```
GET /rest/People/?  
$filter="lastName!=""&$savedfilter="lastName!=""&$orderby="salary"&$savedorderby="salary"&$method=entity  
set
```

Luego, cuando acceda a su conjunto de entidades, escriba lo siguiente (utilizando tanto `$savedfilter` como `$savedorderby`) para asegurarse de que el filtro y su orden de clasificación siempre existen:

```
GET /rest/People/$entityset/AEA452C2668B4F6E98B6FD2A1ED4A5A8?  
$savedfilter="lastName!=""&$savedorderby="salary"
```

# \$skip

Inicia la entidad definida por este número en la colección ( *por ejemplo*, `$skip=10` )

## Descripción

`$skip` define la entidad de la colección por la que se va a comenzar. Por defecto, la colección enviada comienza con la primera entidad. Para comenzar con la décima entidad de la colección, pase 10.

`$skip` se utiliza generalmente junto con `$top/$limit` para navegar por una colección de entidades.

## Ejemplo

En el siguiente ejemplo, vamos a la vigésima entidad de nuestro conjunto de entidades:

```
GET /rest/Employee/$entityset/CB1BCC603DB0416D939B4ED379277F02?$skip=20
```

# \$timeout

Define el número de segundos para guardar un conjunto de entidades en la caché de 4D Server (*por ejemplo, \$timeout=1800*)

## Descripción

Para definir un tiempo de espera para un conjunto de entidades creado con `$method=entityset`, pase el número de segundos a `$timeout`. Por ejemplo, si quiere fijar el tiempo de espera en 20 minutos, pase 1200. Por defecto, el tiempo de espera es de dos (2) horas.

Una vez que se ha definido el tiempo de espera, cada vez que se llama a un conjunto de entidades (mediante el uso de `$method=entityset`), el tiempo de espera se recalcula basándose en la hora actual y el tiempo de espera.

Si se elimina un conjunto de entidades y se vuelve a crear con `$method=entityset` junto con `$savedfilter`, el nuevo tiempo de espera por defecto es de 10 minutos, independientemente del tiempo de espera que haya definido al llamar a `$timeout`.

## Ejemplo

En el conjunto de entidades que estamos creando, definimos el tiempo de espera a 20 minutos:

```
GET /rest/Employee/?$filter="salary!=0"&$method=entityset&$timeout=1200
```

# \$top/\$limit

Limita el número de entidades a devolver (por ejemplo, `$top=50`)

## Descripción

`$top/$limit` define el límite de entidades a devolver. Por defecto, el número está limitado a 100. Puede utilizar las siguientes palabras claves: `$top` o `$limit`.

Cuando se utiliza junto con `$skip`, se puede navegar a través de la selección de entidades devuelta por la petición REST.

## Ejemplo

En el siguiente ejemplo, solicitamos las diez entidades siguientes a la vigésima entidad:

```
GET /rest/Employee/$entityset/CB1BCC603DB0416D939B4ED379277F02?$skip=20&$top=10
```

# \$version

Número de versión de la imagen

## Descripción

`$version` es el número de versión de la imagen devuelto por el servidor. El número de versión, que es enviado por el servidor, funciona en torno a la memoria caché del navegador para que usted esté seguro de recuperar la imagen correcta.

El valor del parámetro de versión de la imagen es modificado por el servidor.

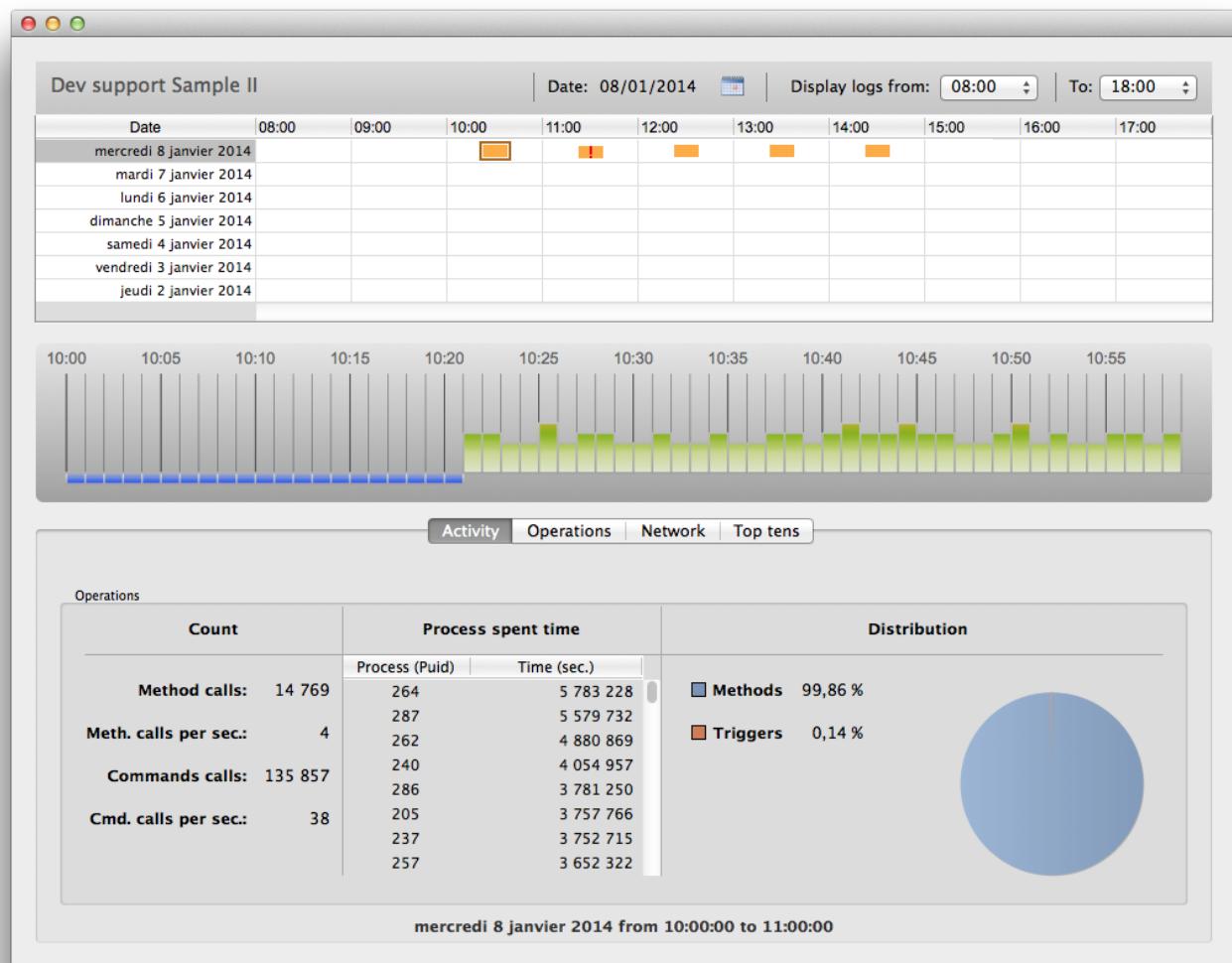
## Ejemplo

El siguiente ejemplo define el formato de la imagen a JPEG independientemente del tipo real de la foto y pasa el número de versión real enviado por el servidor:

```
GET /rest/Employee(1)/photo?$imageformat=jpeg&$version=3&$expand=photo
```

# Acerca de los formularios 4D

Los formularios ofrecen la interfaz a través de la cual se introduce, modifica e imprime la información en una aplicación de escritorio. Los usuarios interactúan con los datos de una base de datos mediante formularios e imprimen informes utilizando formularios. Los formularios pueden utilizarse para crear cajas de diálogo personalizadas, paletas o toda ventana personalizada.



Los formularios también pueden contener otros formularios a través de las siguientes funcionalidades:

- [los objetos subformularios](#)
- [los formularios heredados](#)

## Creación de formularios

Puede añadir o modificar formularios 4D utilizando los siguientes elementos:

- La interfaz 4D Developer: cree nuevos formularios desde el menú Archivo o la ventana del Explorador.
- El editor de formularios: modifique sus formularios utilizando el [editor de formularios](#).
- El código JSON: cree y diseñe sus formularios utilizando JSON y guarde los archivos de los formularios en la [ubicación adecuada](#). Ejemplo:

```
{
    "windowTitle": "Hello World",
    "windowMinWidth": 220,
    "windowMinHeight": 80,
    "method": "HWexample",
    "pages": [
        null,
        {
            "objects": {
                "text": {
                    "type": "text",
                    "text": "Hello World!",
                    "textAlign": "center",
                    "left": 50,
                    "top": 120,
                    "width": 120,
                    "height": 80
                },
                "image": {
                    "type": "picture",
                    "pictureFormat": "scaled",
                    "picture": "/RESOURCES/Images/HW.png",
                    "alignment": "center",
                    "left": 70,
                    "top": 20,
                    "width": 75,
                    "height": 75
                },
                "button": {
                    "type": "button",
                    "text": "OK",
                    "action": "Cancel",
                    "left": 60,
                    "top": 160,
                    "width": 100,
                    "height": 20
                }
            }
        }
    ]
}
```

## Formulario proyecto y formulario tabla

Hay dos categorías de formularios:

- Los formularios de proyecto - Formularios independientes que no están unidos a ninguna tabla. Están pensados, sobre todo, para crear cajas de diálogo de interfaz, al igual que componentes. Los formularios proyecto pueden utilizarse para crear interfaces que cumplan fácilmente con los estándares del sistema operativo.
- Los formularios tablas - Se adjuntan a tablas específicas y, por tanto, se benefician de funciones automáticas útiles para el desarrollo de aplicaciones basadas en bases de datos. Normalmente, una tabla tiene formularios de entrada y salida separados.

Normalmente, se selecciona la categoría del formulario al crearlo, pero se puede cambiar después.

## Páginas formulario

Cada formulario consta de al menos dos páginas:

- una página 1: una página principal, mostrada por defecto

- una página 0: una página de fondo, cuyo contenido se muestra en todas las demás páginas.

Puede crear varias páginas para un formulario de entrada. Si tiene más campos o variables de los que caben en una pantalla, puede crear páginas adicionales para mostrarlos. Las páginas múltiples le permiten hacer lo siguiente:

- Coloque la información más importante en la primera página y la menos importante en otras.
- Organice cada tema en su propia página.
- Reducir o eliminar el desplazamiento durante la entrada de datos definiendo el [orden de entrada](#).
- Deje espacio alrededor de los elementos del formulario para lograr un diseño de pantalla atractivo.

Las páginas múltiples son útiles sólo para los formularios de entrada. No son para imprimir. Cuando se imprime un formulario de varias páginas, sólo se imprime la primera.

No hay restricciones en el número de páginas que puede tener un formulario. El mismo campo puede aparecer un número ilimitado de veces en un formulario y en todas las páginas que desee. Sin embargo, cuantas más páginas tenga un formulario, más tiempo tardará en mostrarse.

Un formulario multipáginas tiene una página de fondo y varias páginas de visualización. Los objetos que se colocan en la página de fondo pueden ser visibles en todas las páginas de visualización, pero sólo se pueden seleccionar y editar en la página de fondo. En los formularios multipágina, debe colocar su paleta de botones en la página de fondo. También es necesario incluir uno o más objetos en la página de fondo que ofrezcan las herramientas de navegación para el usuario.

## Formularios heredados

Los formularios 4D pueden utilizar y ser utilizados como "formularios heredados", lo que significa que todos los objetos de *Formulario A* pueden ser utilizados en *Formulario B*. En este caso, *Formulario B* "hereda" los objetos de *Formulario A*.

Las referencias a un formulario heredado están siempre activas: si se modifica un elemento de un formulario heredado (estilos de botón, por ejemplo), se modificarán automáticamente todos los formularios que utilicen este elemento.

Todos los formularios (formularios tabla y formularios proyecto) pueden ser designados como un formulario heredado. Sin embargo, los elementos que contienen deben ser compatibles con el uso en diferentes tablas de la base de datos.

Cuando se ejecuta un formulario, los objetos se cargan y combinan en el siguiente orden:

1. Página cero del formulario heredado
2. Página 1 del formulario heredado
3. Página cero del formulario abierto
4. Página actual del formulario abierto.

Este orden determina el [orden de entrada](#) de los objetos en el formulario.

Sólo las páginas 0 y 1 del formulario heredado pueden aparecer en otros formularios.

Las propiedades y el método de un formulario no se tienen en cuenta cuando ese formulario se utiliza como formulario heredado. Por otro lado, se llaman los métodos de los objetos que contiene.

Para definir un formulario heredado, las propiedades [Inherited Form Name](#) and [Inherited Form Table](#) (para el formulario tabla) deben definirse en el formulario que heredará algo de otro formulario.

Un formulario puede heredar de un formulario proyecto, definiendo la propiedad [Inherited Form Table](#) en <None> la Lista de propiedades (o " " en JSON).

Para dejar de heredar un formulario, seleccione <none> en la lista de propiedades (o " " en JSON) para la propiedad [Inherited Form Name](#).

Es posible definir un formulario heredado en un formulario que eventualmente se utilizará como formulario heredado para un tercer formulario. La combinación de objetos se realiza de forma recursiva. 4D detecta los bucles recursivos (por ejemplo, si el formulario [table1]form1 se define como el formulario heredado de [table1]form1, es decir, él mismo) e interrumpe la cadena de formularios.

## Propiedades soportadas

[Associated Menu Bar](#) - [Fixed Height](#) - [Fixed Width](#) - [Form Break](#) - [Form Detail](#) - [Form Footer](#) - [Form Header](#) - [Form Name](#) - [Form Type](#) - [Inherited Form Name](#) - [Inherited Form Table](#) - [Maximum Height](#) - [Maximum Width](#) - [Method](#) - [Minimum Height](#) - [Minimum Width](#) - [Pages](#) - [Print Settings](#) - [Published as Subform](#) - [Save Geometry](#) - [Window Title](#)

# Editor de formularios

4D ofrece un completo editor de formularios que le permite modificar su formulario hasta conseguir el efecto que desea. Con el editor de formularios puede crear y eliminar objetos formulario, manipularlos directamente y definir las propiedades de los mismos.

## Interface

La interfaz del editor de formularios muestra cada formulario JSON en su propia ventana, que tiene una barra de herramientas y otra de objetos. Puede tener varios formularios abiertos al mismo tiempo.

### Opciones de visualización

You can show or hide several interface elements on the current page of the form:

- Inherited Form: Inherited form objects (if there is an [inherited form](#)).
- Page 0: Objects from [page 0](#). Esta opción permite distinguir entre los objetos de la página actual del formulario y los de la página 0.
- Paper: Borders of the printing page, which are shown as gray lines. This element can only be displayed by default in "for printing" type forms.
- Rulers: Rulers of the Form editor's window.
- Markers: Output control lines and associated markers that show the limits of the form's different areas. This element can only be displayed by default in [list forms](#).
- Etiquetas de los marcadores : etiquetas de los marcadores, disponibles sólo cuando se muestran las líneas de control de salida. This element can only be displayed by default in [list forms](#).
- Límites: Límites del formulario. Cuando se selecciona esta opción, el formulario se muestra en el editor de formularios tal y como aparece en el modo Aplicación. De esta manera puede ajustar su formulario sin tener que cambiar al modo Aplicación para ver el resultado.

The [Size Based on](#), [Hor. margin](#) and [Vert. margin](#) settings of the form properties affect the form's limits.

Cuando se utilizan estos parámetros, los límites se basan en los objetos del formulario. Cuando se modifica el tamaño de un objeto que se encuentra junto al límite del formulario, el rectángulo de delimitación se modifica para reflejar ese cambio.

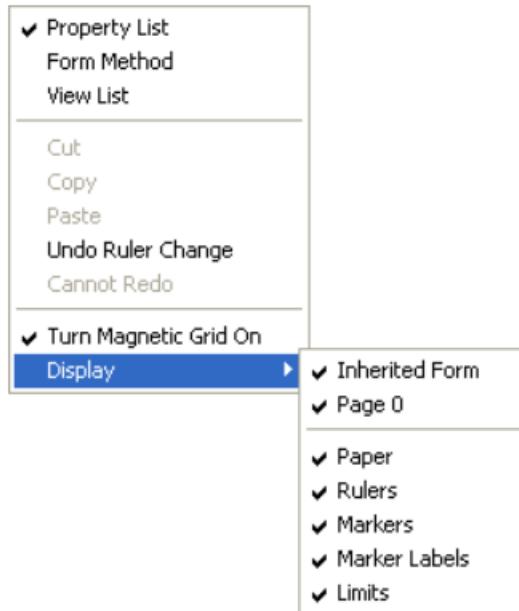
### Visualización por defecto

When a form is opened in the editor, interface elements are displayed or hidden by default, depending on:

- the New form default display options set in the Preferences - unchecked options cannot be displayed by default.
- the current [form type](#):
  - Markers and marker labels are always displayed by default on list forms
  - Paper is displayed by default on "for printing" forms.

### Mostrar/Ocultar elementos

You can display or hide elements at any moment in the Form editor's current window by selecting [Display](#) from the Form menu or the Form editor's context menu:



## Reglas

Las reglas laterales e inferiores le ayudan a posicionar los objetos en el formulario. Pueden [mostrarse u ocultarse](#).

Select Ruler definition... from the Form menu to change measurement units so that the form displays inches, centimeters, or pixels.

## Barra de herramientas

La barra de herramientas del editor de formularios ofrece un conjunto de herramientas para manipular y modificar el formulario. Cada ventana tiene su propia barra de herramientas.



La barra de herramientas contiene los siguientes elementos:

Icono	Nombre	Descripción
	Ejecutar el formulario	Se utiliza para probar la ejecución del formulario. Al presionar este botón, 4D abre una nueva ventana y muestra el formulario en su contexto (lista de registros para un formulario lista y página de registro actual para un formulario detallado). El formulario se ejecuta en el proceso principal.
	[[[Herramienta de selección]]] [#seleccionar - objetos]] [#seleccionar - objetos]] [#seleccionar - objetos])	Permite seleccionar, mover y redimensionar los objetos del formulario. Nota: cuando se selecciona un objeto de tipo Texto o Área de Grupo, al presionar la tecla Intro se pasa al modo de edición.
	Orden de entrada	Pasa al modo "Orden de entrada", donde es posible ver y cambiar el orden de entrada actual del formulario. Tenga en cuenta que las marcas permiten ver el orden de entrada actual, sin dejar de trabajar en el formulario.
	Desplazamiento	Pasa al modo "Desplazamiento", en el que es posible llegar rápidamente a cualquier parte del formulario utilizando la función de arrastrar y soltar en la

Icono	Nombre	Descripción
	Zoom	Permite modificar la escala de visualización del formulario (100% por defecto). Puede pasar al modo "Zoom" haciendo clic en la lupa o pulsando directamente en la barra correspondiente a la escala deseada. Esta función se detalla en la sección anterior.
	Alignement	Este botón está asociado a un menú que permite alinear los objetos en el formulario. Se activa (o no) en función de los objetos seleccionados. Disponible sólo con CSS Preview None
	Distribución	Este botón está asociado a un menú que permite repartir los objetos en el formulario. Se activa (o no) en función de los objetos seleccionados. Disponible sólo con CSS Preview None
	Nivel	Este botón está asociado a un menú que permite cambiar el nivel de los objetos en el formulario. Se activa (o no) en función de los objetos seleccionados.
	Agrupar/desagrupar	Este botón está asociado a un menú que permite agrupar y desagrupar la selección de objetos del formulario. Se activa (o no) en función de los objetos seleccionados.
	Visualización y gestión de páginas	Esta área permite pasar de una página de formulario a otra y añadir páginas. Para navegar entre las páginas del formulario, haga clic en los botones de flecha o en el área central y elija la página que desea visualizar en el menú que aparece. Si pulsa el botón flecha derecha mientras se muestra la última página del formulario, 4D le permite añadir una página.
	Vista previa del CSS	Este botón se utiliza para seleccionar el modo CSS a utilizar.
	Gestión de vistas	Este botón muestra u oculta la paleta de vistas. Esta función se detalla en la sección Utilizar las vistas de objeto.
	Visualización de marcas	Cada clic en este botón provoca la visualización sucesiva de todos los tipos de escudos de formulario. El botón también está vinculado a un menú que permite seleccionar directamente el tipo de escudo a mostrar.
	Librería de objetos preconfigurada	Este botón muestra la librería de objetos preconfigurada que ofrece numerosos objetos con ciertas propiedades que han sido predefinidas.
	Creación de list box	Este botón crea nuevos list box de tipo selección de entidades.

## Barra de objetos

The object bar contains all the active and inactive objects that can be used in 4D forms. Algunos objetos se agrupan por temas. Each theme includes several alternatives that you can choose between. When the object bar has the focus, you can select the buttons using the keys of the keyboard. The following table describes the object groups available and their associated shortcut key.

Botón	Agrupar	Llave
	Text / Group Box	T
	Entrada	F
	Hierarchical List / List Box	L
	Combo Box / Drop-down List / Picture Pop-up Menu	P
	Button / Picture Button / Button Grid	B
	Botón radio	R
	Casilla a seleccionar	C
	Progress Indicator / Ruler / Stepper / Spinner	I
	Rectangle / Line / Oval	S
	Splitter / Tab Control	D
	Plug-in Area / Subform / Web Area / 4D Write Pro / 4D View Pro	X

To draw an object type, select the corresponding button and then trace the object in the form. After creating an object, you can modify its type using the Property List. Hold down the Shift key as you draw to constrain the object to a regular shape. Lines are constrained to horizontal, 45°, or vertical, rectangles are constrained to squares, and ovals are constrained to circles.

The current variant of the theme is the object that will be inserted in the form. When you click the right side of a button, you access the variant menu:



You can click twice on the button so that it remains selected even after you have traced an object in the form (continual selection). Esta función facilita la creación de varios objetos sucesivos del mismo tipo. Para cancelar una selección continua, haga clic en otro objeto o herramienta.

## Lista de propiedades

Both forms and form objects have properties that control access to the form, the appearance of the form, and the behavior of the form when it is used. Form properties include, for example, the form's name, its menu bar, and its size. Object Properties include, for example, an object's name, its dimensions, its background color, and its font.

Puede visualizar y modificar las propiedades de los objetos y formularios utilizando la lista de propiedades. Muestra las propiedades del formulario o de los objetos en función de lo que se seleccione en la ventana del editor.

To display/hide the Property List, choose Property List from the Form menu or from the context menu of the Form editor. También puede mostrarlo haciendo doble clic en una área vacía del formulario.

## Accesos directos de navegación

Puede navegar en la Lista de Propiedades utilizando los siguientes atajos:

- Tecla de flechas ↑ ↓ : se utiliza para pasar de una celda a otra.

- Teclas flechas  $\leftarrow \rightarrow$ : se utiliza para expandir/contraer los temas o entrar en el modo de edición.
- PgUp y PgDn: selecciona la primera o la última celda visible de la lista mostrada.
- Inicio y Fin: selecciona la primera o la última celda de la lista.
- Ctrl+clic (Windows) o Comando+clic (Mac OS) en un evento: selecciona/deselecciona todos los eventos de la lista, en función del estado inicial del evento sobre el que se ha hecho clic.
- Ctrl+clic (Windows) o Comando+clic (Mac OS) en la etiqueta de tema: contrae/despliega todos los temas de la lista.

## Manipulación de objetos formulario

### Añadir objetos

Puede añadir objetos en los formularios de varias maneras:

- Dibujando el objeto directamente en el formulario después de seleccionar su tipo en la barra de objetos (ver [Utilizando la barra de objetos](#))
- Arrastrando y soltando el objeto desde la barra de objetos
- Mediante operaciones de arrastrar y soltar o copiar y pegar sobre un objeto seleccionado de la [librería de objetos](#) preconfigurada,
- Arrastrando y soltando un objeto desde otro formulario,
- Arrastrando y soltando un objeto desde el Explorador (campos) o desde otros editores del modo Diseño (listas, imágenes, etc.)

Una vez insertado el objeto en el formulario, puede modificar sus características utilizando el editor de formularios.

Puede trabajar con dos tipos de objetos en sus formularios:

- Objetos estáticos (líneas, marcos, imágenes de fondo, etc.): estos objetos se utilizan generalmente para definir la apariencia del formulario y sus etiquetas, así como para la interfaz gráfica. Están disponibles en la barra de objetos del editor de formularios. También puede definir sus atributos gráficos (tamaño, color, fuente, etc.) y sus propiedades de redimensionamiento utilizando la lista de propiedades. Los objetos estáticos no tienen variables asociadas como los objetos activos. Sin embargo, se pueden insertar objetos dinámicos en objetos estáticos.
- Objetos activos: estos objetos realizan tareas o funciones en la interfaz y pueden adoptar muchas formas: campos, botones, listas desplazables, etc. Cada objeto activo está asociado a un campo o a una variable.

### Seleccionar los objetos

Antes de poder realizar cualquier operación en un objeto (como cambiar el ancho de la línea o la fuente), debe seleccionar el objeto que desea modificar.

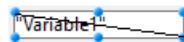
Para seleccionar un objeto utilizando la barra de herramientas:

1. Haga clic en la herramienta Flecha de la barra de herramientas.



Cuando se mueve el puntero en el área del formulario, se convierte en un puntero estándar con forma de flecha.

2. Haga clic en el objeto que desea seleccionar. Las manillas de redimensionamiento identifican el objeto seleccionado.



Para seleccionar un objeto utilizando la Lista de propiedades:

1. Seleccione el nombre del objeto en la lista desplegable de objetos situada en la parte superior de la lista de propiedades.

Con estos dos métodos, puede seleccionar un objeto que esté oculto por otros objetos o que se encuentre fuera del área visible de la ventana actual. Para deseleccionar un objeto, haga clic fuera del límite del objeto o Mayúsculas+clic en el objeto.

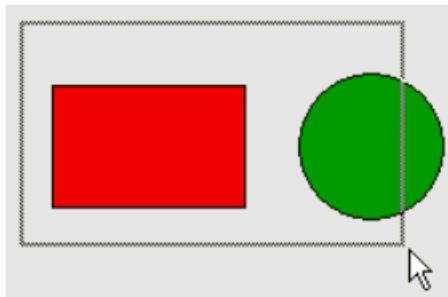
También es posible seleccionar objetos haciendo doble clic en la ventana de resultados de la operación "Buscar en diseño".

## Selección de múltiples objetos

Es posible que desee realizar la misma operación en más de un objeto de un formulario, por ejemplo, para mover los objetos, alinearlos o cambiar su apariencia. 4D le permite seleccionar varios objetos al mismo tiempo. Hay varias formas de seleccionar varios objetos:

- Seleccione Seleccionar todo en el menú Edición para seleccionar todos los objetos.
- Haga clic con el botón derecho en el objeto y elija el comando Seleccionar objetos del mismo tipo en el menú contextual.
- Mantenga presionada la tecla Mayús y haga clic en los objetos que desee seleccionar.
- Comience en una ubicación fuera del grupo de objetos que desea seleccionar y arrastre un marco (a veces llamado rectángulo de selección) alrededor de los objetos. Al soltar el botón del ratón, si alguna parte de un objeto se encuentra dentro o toca los límites del rectángulo de selección, ese objeto queda seleccionado.
- Mantenga presionada la tecla Alt (Windows) o la tecla Opción (macOS) y dibuje un rectángulo de selección. Se selecciona todo objeto que esté completamente encerrado por el marco.

La imagen siguiente muestra el dibujo de un rectángulo para seleccionar dos objetos:



Para deseleccionar un objeto que forma parte de un grupo de objetos seleccionados, mantenga presionada la tecla Mayús y haga clic en el objeto. Los demás objetos permanecen seleccionados. Para deseleccionar todos los objetos seleccionados, haga clic fuera de los límites de todos los objetos.

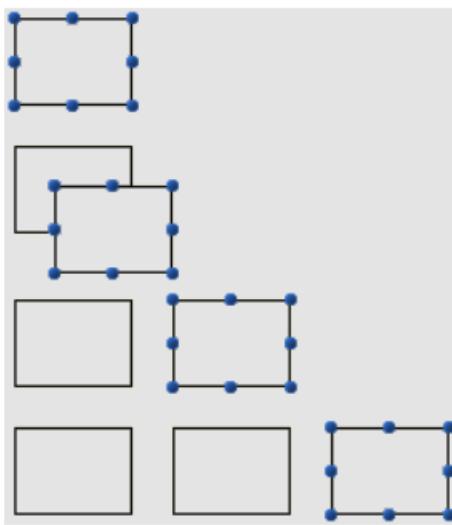
## Duplicar los objetos

Puede duplicar todo objeto de formulario, incluidos los objetos activos. Las copias de los objetos activos conservan todas las propiedades del objeto original, incluidos el nombre, el tipo, la acción estándar, el formato de visualización y el método objeto.

Puede duplicar un objeto directamente con la herramienta Duplicar de la paleta Herramientas o utilizar la caja de diálogo Duplicar varios para duplicar un objeto más de una vez. Además, a través de esta caja de diálogo, se puede definir la distancia entre dos copias.

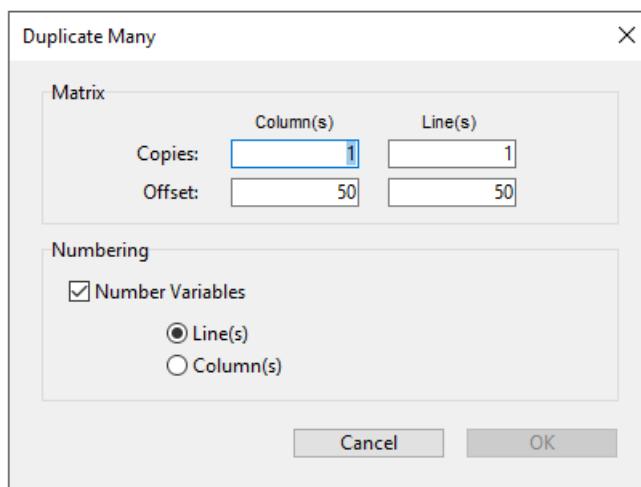
Para duplicar uno o más objetos:

1. Seleccione el objeto u objetos que desea duplicar.
2. Elija Duplicar en el menú Edición. 4D crea una copia de cada objeto seleccionado y coloca la copia delante y justo al lado del original.
3. Mueva la copia (o las copias) a la ubicación deseada. Si vuelve a elegir el elemento de menú Duplicar, 4D crea otra copia de cada objeto y la mueve exactamente a la misma distancia y dirección de la primera copia. Si necesita distribuir copias del objeto a lo largo de una línea, debe utilizar el siguiente procedimiento. Duplique el objeto original, mueva la copia a otro lugar del formulario y luego duplique la copia. La segunda copia se coloca automáticamente en la misma posición que la primera copia tenía en relación con el objeto original. Las copias posteriores también se sitúan en la misma relación con sus originales. La siguiente figura explica el funcionamiento de la ubicación relativa de las copias:



## Duplicar muchos

La caja de diálogo "Duplicar muchos" aparece cuando se selecciona uno o más objetos y se elige el comando Duplicar muchos... del menú Objeto.



- En el área superior, introduzca el número de columnas y líneas de objetos que desea obtener.

Por ejemplo, si desea tres columnas y dos líneas de objetos, introduzca 3 en el área Columna(s) y 2 en el área Línea(s). Si desea tres nuevas copias horizontales de un objeto, introduzca 4 en el área Columna(s) y deje el valor por defecto, 1, en el área Línea(s).

- Para las líneas y columnas, defina el desplazamiento que desea aplicar a cada copia.

El valor debe expresarse en puntos. Se aplicará a cada copia, o copias, en relación con el objeto original.

Por ejemplo, si desea dejar un desplazamiento vertical de 20 puntos entre cada objeto y la altura del objeto fuente es de 50 puntos, introduzca 70 en el área "Intervalo" de la columna.

- Si desea crear una matriz de variables, seleccione la opción Numerar las variables y seleccione la dirección en la que se van a numerar las variables, ya sea por línea(s) o por columna(s). Esta opción sólo se activa cuando el objeto seleccionado es una variable. Para más información sobre esta opción, consulte Duplicar en una matriz en el *Manual de diseño*.

## Mover objetos

Puede mover todo gráfico u objeto activo del formulario, incluidos los campos y los objetos creados con una plantilla. Al mover un objeto, tiene las siguientes opciones:

- Mover el objeto arrastrándolo,
- Mover el objeto un píxel a la vez utilizando las teclas de flecha,
- Mover el objeto por pasos utilizando las teclas de flecha (pasos de 20 píxeles por defecto),

Al comenzar a arrastrar el objeto seleccionado, sus controles desaparecen. 4D muestra marcadores que indican la ubicación de los límites del objeto en las reglas para que pueda colocar el objeto exactamente donde lo quiere. Tenga cuidado de no arrastrar un mango. Al arrastrar un mango se cambia el tamaño del objeto. Puede presionar la tecla Mayúsculas para realizar el movimiento con una restricción.

Cuando la [rejilla magnética](#) está activada, los objetos se mueven por etapas indicando ubicaciones perceptibles.

Para mover un objeto un píxel por píxel:

- Seleccione el objeto u objetos y utilice las teclas de flecha del teclado para mover el objeto. Cada vez que se presiona una tecla flecha, el objeto se mueve un píxel en la dirección de la flecha.

Mover un objeto por pasos:

- Selecciona el objeto u objetos que quiera mover y mantenga presionada la tecla Mayúsculas y utilice las teclas de dirección para mover el objeto por pasos. Por defecto, los pasos son de 20 píxeles cada vez. Puede cambiar este valor en la página Formularios de las Preferencias.

## Agrupar objetos

4D le permite agrupar objetos para que pueda seleccionar, mover y modificar el grupo como un solo objeto. Los objetos agrupados conservan su posición en relación con los demás. Lo normal es agrupar un campo y su etiqueta, un botón invisible y su ícono, etc.

Cuando se redimensiona un grupo, todos los objetos del grupo se redimensionan proporcionalmente (excepto las áreas de texto, que se redimensionan por pasos según el tamaño de sus fuentes).

Puede desagrupar un grupo de objetos para tratarlos de nuevo como objetos individuales.

Un objeto activo que ha sido agrupado debe ser desagrupado antes de poder acceder a sus propiedades o métodos. Sin embargo, es posible seleccionar un objeto perteneciente a un grupo sin reagrupar el conjunto: para ello, Ctrl+clic (Windows) o Comando+clic (macOS) en el objeto (el grupo debe estar seleccionado previamente).

La agrupación sólo afecta a los objetos en el editor de formularios. Cuando se ejecuta el formulario, todos los objetos agrupados actúan como si estuvieran desagrupados.

La rejilla magnética también influye en el redimensionamiento manual de los objetos.

Para agrupar los objetos:

1. Seleccione los objetos que desea agrupar.
2. Elija Agrupar en el menú Objetos.

O

Haga clic en el botón Agrupar de la barra de herramientas del editor de formularios:



4D marca el límite de los objetos recién agrupados con marcas. No hay marcas que delimiten ninguno de los objetos individuales del grupo. Ahora, al modificar el objeto agrupado, se modifican todos los objetos que componen el grupo.

Para desagrupar un grupo de objetos:

1. Seleccione el grupo de objetos que desea desagrupar.
2. Elija Desagrupar en el menú Objetos.

O

Haga clic en el botón Desagrupar (menú del botón Agrupar) de la barra de herramientas del editor de formularios.

Si Desagrupar está atenuado, significa que el objeto seleccionado ya está separado en su forma más simple.

4D marca los bordes de los objetos individuales con marcas.

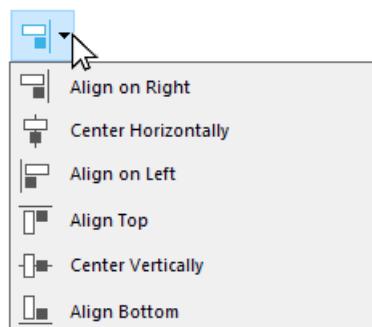
## Alinear objetos

Puede alinear los objetos entre sí o mediante una rejilla invisible en el formulario.

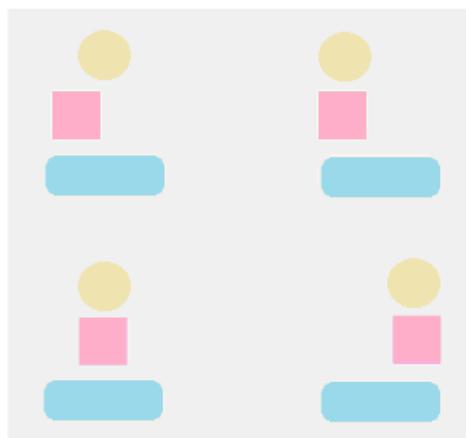
- Cuando se alinea un objeto con otro, se puede alinear con la parte superior, inferior, lateral o con el centro horizontal o vertical del otro objeto. Puede alinear directamente una selección de objetos utilizando las herramientas de alineación o aplicar ajustes de alineación más avanzados utilizando el Asistente de Alineación. Esta última opción permite, por ejemplo, definir el objeto que se utilizará como referencia de posición y previsualizar la alineación en el formulario antes de aplicarla.
- Cuando se utiliza la rejilla invisible, cada objeto puede alinearse manualmente con otros basándose en posiciones "perceptibles" que se representan con líneas de puntos que aparecen cuando el objeto que se mueve se acerca a otros objetos.

### Utilizar las herramientas de alineación directa

Las herramientas de alineación de la barra de herramientas y del submenú Alinear del menú Objeto permiten alinear rápidamente los objetos seleccionados.

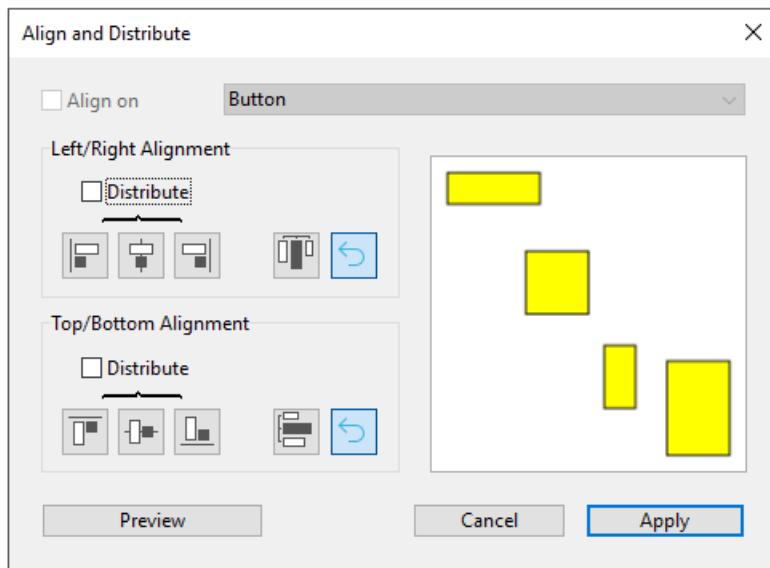


Cuando 4D alinea los objetos, deja un objeto seleccionado en su lugar y alinea el resto de los objetos a ese. This object is the "anchor." Utiliza el objeto que está más lejos en la dirección de la alineación como ancla y alinea los otros objetos a ese objeto. Por ejemplo, si quiere realizar una alineación a la derecha en un conjunto de objetos, el objeto más a la derecha se utilizará como ancla. La figura siguiente muestra objetos sin alineación, "alineados a la izquierda", "alineados horizontalmente por centros" y "alineados a la derecha":



### Utilizar el asistente de alineación

El Asistente de Alineación permite realizar cualquier tipo de alineación y/o distribución de objetos.



Para mostrar esta caja de diálogo, seleccione los objetos que desee alinear y, a continuación, elija el comando Alineación del submenú Alinear del menú Objeto o del menú contextual del editor.

- En las áreas "Alineación izquierda/derecha" y/o "Alineación superior/inferior", haga clic en el icono que corresponde a la alineación que desea realizar.

El área de ejemplo muestra los resultados de su selección.

- Para realizar una alineación que utilice el esquema de anclaje estándar, haga clic en Ver o Aplicar.

En este caso, 4D utiliza el objeto que está más lejos en la dirección de la alineación como ancla y alinea los otros objetos a ese objeto. Por ejemplo, si quiere realizar una alineación a la derecha en un conjunto de objetos, el objeto más a la derecha se utilizará como ancla.

O:

Para alinear los objetos a un objeto específico, seleccione la opción Alinear en y seleccione el objeto al que desea que se alineen los demás objetos de la lista de objetos. En este caso, la posición del objeto de referencia no se alterará.

Puede previsualizar los resultados de la alineación haciendo clic en el botón Previsualización. Los objetos se alinean entonces en el editor de formularios, pero como la caja de diálogo permanece en el primer plano, aún puede cancelar o aplicar la alineación.

Esta caja de diálogo le permite alinear y distribuir objetos en una sola operación. Para más información sobre cómo distribuir objetos, consulte [Repartir objetos](#).

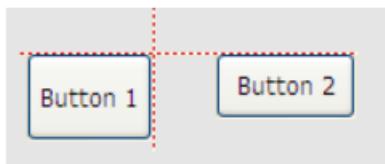
## Utilizar la rejilla magnética

El editor de formularios dispone de una rejilla magnética virtual que puede ayudarle a colocar y alinear objetos en un formulario. La alineación magnética de los objetos se basa en su posición con respecto a los demás. La rejilla magnética sólo puede utilizarse cuando hay al menos dos objetos en el formulario.

Esto funciona de la siguiente manera: cuando mueve un objeto en el formulario, 4D indica las posibles ubicaciones de este objeto basándose en alineaciones notables con otros objetos formulario. Una alineación evidente se establece cada vez que:

- Horizontalmente, los bordes o centros de dos objetos coinciden,
- Verticalmente, los bordes de dos objetos coinciden.

Cuando esto ocurre, 4D coloca el objeto en el lugar y muestra una línea roja que indica la alineación notable que se ha tenido en cuenta:



En cuanto a la distribución de los objetos, 4D ofrece una distancia basada en los estándares de interfaz. Al igual que en el caso de la alineación magnética, las líneas rojas indican las diferencias notables una vez alcanzadas.



Este funcionamiento se aplica a todos los tipos de objetos de los formularios. La rejilla magnética puede activarse o desactivarse en cualquier momento utilizando el comando Activar la rejilla magnética del menú Formulario o del menú contextual del editor. También es posible definir la activación de esta función por defecto en la página Preferencias > Formularios (opción Activar la alineación automática por defecto). Puede activar o desactivar manualmente la rejilla magnética cuando se selecciona un objeto presionando la tecla Ctrl (Windows) o Control (macOS).

Cuando se superponen varios objetos, se puede utilizar el atajo `Ctrl+Mayús+clic` / `Comando+Mayús+clic` para seleccionar cada objeto sucesivamente bajando un plano con cada clic.

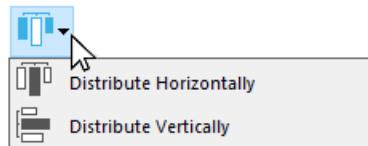
## Distribuir los objetos

Puede repartir los objetos de manera que queden dispuestos con el mismo espacio entre ellos. Para ello, puede distribuir los objetos utilizando las herramientas de distribución de la paleta Herramientas o el Asistente de alineación. Este último le permite alinear y distribuir objetos en una sola operación.

Puede cambiar el orden de entrada en tiempo de ejecución utilizando los comandos `FORM SET ENTRY ORDER` y `FORM GET ENTRY ORDER`.

Para repartir los objetos con igual espacio:

1. Seleccione tres o más objetos y haga clic en la herramienta Distribuir correspondiente.
2. En la barra de herramientas, haga clic en la herramienta de distribución que corresponde a la distribución que desea aplicar.



O

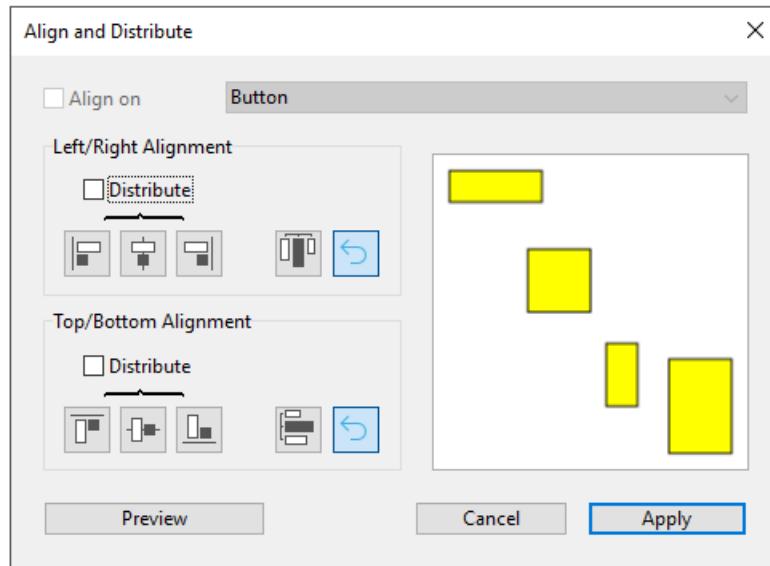
Seleccione un comando del menú de distribución en el submenú Alinear del menú Objeto o en el menú contextual del editor.

4D distribuye los objetos consecuentemente. Los objetos se distribuyen utilizando la distancia a sus centros y se utiliza como referencia la mayor distancia entre dos objetos consecutivos.

Para distribuir objetos utilizando la caja de diálogo Alinear y Distribuir:

1. Seleccione los objetos que desea distribuir.
2. Seleccione el comando Alineación del submenú Alinear del menú Objeto o del menú contextual del editor.

Aparece la siguiente caja de diálogo:



3. En las áreas Alineación izquierda/derecha y/o Alineación superior/inferior, haga clic en el ícono de distribución estándar:



(Ícono de distribución horizontal estándar)

El área de ejemplo muestra los resultados de su selección.

4. Para efectuar una repartición estándar que utilice el esquema estándar, haga clic en Previsualización o Aplicar.

En este caso, 4D realizará una distribución estándar, de modo que los objetos se dispongan con la misma cantidad de espacio entre ellos.

O:

Para efectuar una distribución específica, seleccione la opción Distribuir (por ejemplo, si desea distribuir los objetos en función de la distancia a su lado derecho). Esta opción actúa como un interruptor. Si la casilla de selección Distribuir está seleccionada, los iconos situados debajo de ella realizan una función diferente:

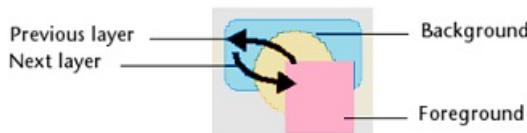
- Horizontalmente, los iconos corresponden a las siguientes distribuciones: uniformemente con respecto a los lados izquierdo, central (hor.) y derecho de los objetos seleccionados.
- Verticalmente, los iconos corresponden a las siguientes distribuciones: uniformemente con respecto a los bordes superiores, centros (vert.) y bordes inferiores de los objetos seleccionados.

Puede previsualizar el resultado real de sus parámetros haciendo clic en el botón Previsualización: la operación se lleva a cabo en el editor de formularios, pero la caja de diálogo permanece en primer plano. Puede entonces Cancelar o Aplicar las modificaciones.

Esta caja de diálogo permite combinar la alineación y la distribución de objetos. Para más información sobre la alineación, consulte [Alinear objetos](#).

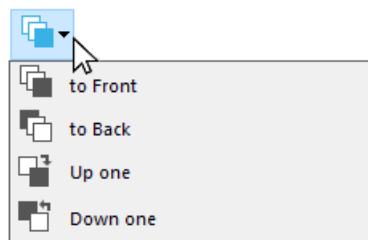
## Gestionar los planos de los objetos

A veces tendrá que reorganizar los objetos que obstruyen la vista de otros objetos del formulario. Por ejemplo, puede tener un gráfico que deseé que aparezca detrás de los campos en un formulario. 4D ofrece cuatro elementos de menú, Mover hacia atrás, Mover hacia delante, Subir un nivel y Bajar un nivel que le permiten organizar los planos de los objetos en el formulario. Estas capas también determinan el orden de entrada por defecto (ver Modificación del orden de entrada de datos). La figura siguiente muestra objetos delante y detrás de otros objetos:



Para mover un objeto a otro plano, selecciónelo y elija:

- Uno de los comandos Mover hacia atrás, Mover hacia delante, Subir un nivel y Bajar un nivel en el menú Objeto,
- Uno de los comandos del submenú Plano> del menú contextual del editor,
- Uno de los comandos asociados al botón de gestión de los planos de la barra de herramientas.



Cuando se superponen varios objetos, se puede utilizar el atajo Ctrl+Mayús+clic / Comando+Mayús+clic para seleccionar cada objeto sucesivamente bajando un plano con cada clic.

Al ordenar los diferentes niveles, 4D siempre va del fondo al primer plano. Como resultado, el nivel anterior desplaza la selección de objetos de un plano hacia el fondo del formulario. El siguiente nivel mueve la selección un plano hacia el primer plano del formulario.

## Orden de entrada de los datos

El orden de entrada de datos es el orden en el que se seleccionan los campos, subformularios y otros objetos activos al presionar la tecla Tab o la tecla Retorno de carro en un formulario de entrada. Es posible desplazarse por el formulario en sentido contrario (invertir el orden de entrada de datos) presionando las teclas Mayús+Tab o Mayús+Retorno de carro de retorno.

Puede cambiar el orden de entrada en tiempo de ejecución utilizando los comandos `FORM SET ENTRY ORDER` y `FORM GET ENTRY ORDER`.

Todos los objetos que soportan la propiedad enfocable se incluyen por defecto en el orden de entrada de datos.

La configuración del orden de entrada para un formulario JSON se realiza con la propiedad `entryOrder`.

Si no especifica un orden de entrada personalizado, 4D utiliza por defecto el plano de los objetos para determinar el orden de entrada en el sentido "fondo hacia primer plano." El orden de entrada estándar corresponde, por tanto, al orden de creación de los objetos en el formulario.

En algunos formularios, se necesita definir un orden de entrada de datos personalizado. A continuación, por ejemplo, se han añadido campos adicionales relacionados con la dirección después de la creación del formulario. El orden de entrada estándar resultante se vuelve ilógico y obliga al usuario a introducir la información de forma extraña:

En casos como éste, un orden de entrada de datos personalizado le permite introducir la información en un orden más lógico:

## Visualizar y modificar el orden de entrada de datos

Puede ver el orden de entrada actual utilizando marcas "Orden de entrada" o utilizando el modo "Orden de entrada". Sin embargo, sólo puede modificar el orden de entrada utilizando el modo "Orden de entrada".

Este párrafo describe la visualización y la modificación del orden de entrada en modo "Orden de entrada". Para más información sobre la visualización del orden de entrada utilizando marcas, consulte [Using shields](#).

Para ver o cambiar el orden de entrada:

1. Seleccione Orden de entrada en el menú Formulario o haga clic en el botón Orden de entrada en la barra de herramientas de la ventana:



El puntero se convierte en un puntero de orden de entrada y 4D dibuja una línea en el formulario mostrando el orden en que selecciona los objetos durante la entrada de datos.

Ver y cambiar el orden de entrada de datos son las únicas acciones que puede realizar hasta que haga clic en cualquier herramienta de la paleta Herramientas.

2. Para cambiar el orden de entrada de datos, sitúe el puntero sobre un objeto del formulario y, mientras mantiene presionado el botón del ratón, arrastre el puntero hasta el objeto que desee que siga en el orden de entrada de datos.

4D ajustará la orden de entrada en consecuencia.

3. Repita el paso 2 tantas veces como sea necesario para establecer el orden de entrada de datos que deseé.
4. Cuando esté satisfecho con el orden de entrada de datos, haga clic en cualquier herramienta no seleccionada de la barra de herramientas o elija Orden de entrada en el menú Formulario.

4D vuelve al modo de funcionamiento normal del editor de formularios.

Sólo se muestra el orden de entrada de la página actual del formulario. Si el formulario contiene objetos editables en la página 0 o procedentes de un formulario heredado, el orden de entrada por defecto es el siguiente: objetos de la página 0 del formulario heredado > objetos de la página 1 del formulario heredado > objetos de la página 0 del formulario abierto > objetos de la página actual del formulario abierto.

## Utilizar un grupo de entrada

Cuando se cambia el orden de entrada, se puede seleccionar un grupo de objetos en el formulario para que el orden de entrada estándar se aplique a los objetos del grupo. Esto le permite definir fácilmente el orden de entrada para los formularios en los que los campos están separados en grupos o columnas.

Para crear un grupo de entrada:

1. Seleccione Orden de entrada en el menú *Formulario* o haga clic en el botón de la barra de herramientas.
2. Dibuje un rectángulo de selección alrededor de los objetos que desee agrupar para la entrada de datos.

Al soltar el botón del ratón, los objetos encerrados o tocados por el rectángulo siguen el orden de entrada por defecto. El orden de entrada de los otros objetos restantes se ajusta según sea necesario.

## Excluir un objeto del orden de entrada

Por defecto, todos los objetos que soportan la propiedad enfocable se incluyen en el orden de entrada. Para excluir un objeto del orden de entrada:

1. Seleccione el modo de orden de entrada, y luego
2. shift-click on the object
3. right-click on the object and select Remove from entry order option from the context menu

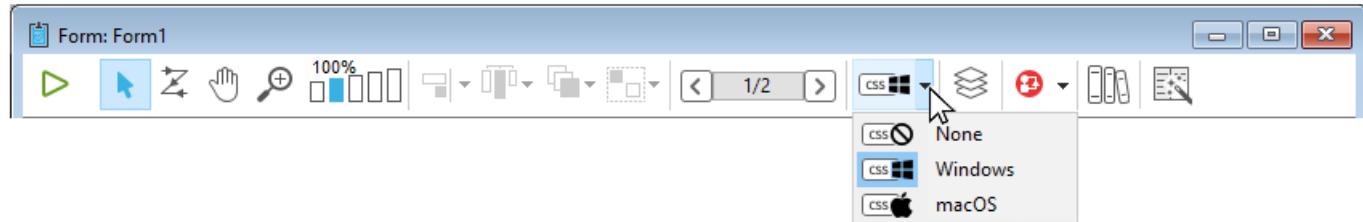
## Vista previa del CSS

The Form editor allows you to view your forms with or without applied CSS values.

When [style sheets](#) have been defined, forms (including inherited forms and subforms) are opened in the CSS Preview mode for your operating system by default.

## Seleccionando el modo de vista previa CSS

The Form editor toolbar provides a CSS button for viewing styled objects:



Select one of the following preview modes from the menu:

Icono barra de herramientas	Modo de vista previa CSS	Descripción
	Ninguno	No CSS values are applied in the form and no CSS values or icons displayed in the Property List.
	Windows	CSS values for Windows platform are applied in the form. CSS values and icons displayed in the Property List.
	macOS	CSS values for macOS platform are applied in the form. CSS values and icons displayed in the Property List.

If a font size too large for an object is defined in a style sheet or JSON, the object will automatically be rendered to accommodate the font, however the size of the object will not be changed.

The CSS preview mode reflects the priority order applied to style sheets vs JSON attributes as defined in the [JSON vs Style Sheet](#) section.

Once a CSS preview mode is selected, objects are automatically displayed with the styles defined in a style sheet (if any).

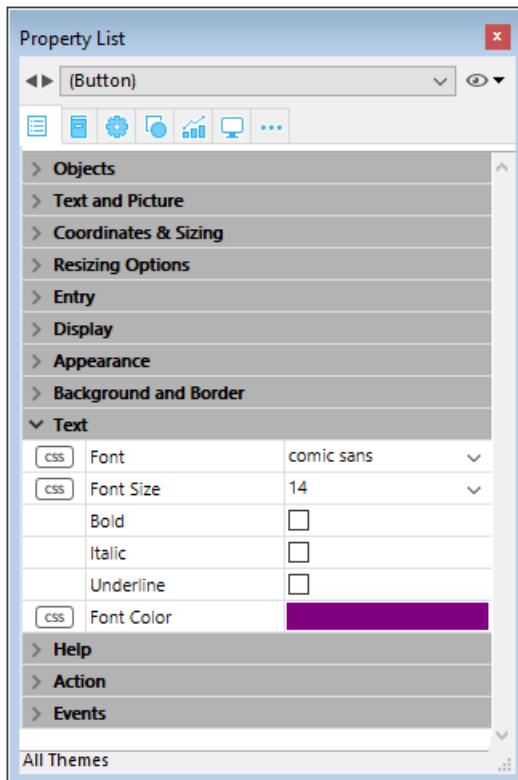
When copying or duplicating objects, only the CSS references (if any) and the JSON values are copied.

## Soporte CSS en la Lista de Propiedades

In CSS Preview mode, if the value of an attribute has been defined in a style sheet, the attribute's name will appear with a CSS icon displayed next to it in the Property List. For example, the attribute values defined in this style sheet:

```
.myButton {  
    font-family: comic sans;  
    font-size: 14;  
    stroke: #800080;  
}
```

are displayed with a CSS icon in the Property List:



An attribute value defined in a style sheet can be overridden in the JSON form description (except if the CSS includes the `!important` declaration, see below). In this case, the Property List displays the JSON form value in bold. You can reset the value to its style sheet definition with the Ctrl + click (Windows) or Command + click (macOs) shortcuts.

If an attribute has been defined with the `!important` declaration for a group, an object within a group, or any object within a selection of multiple objects, that attribute value is locked and cannot be changed in the Property List.

## Lista de propiedades de iconos CSS

Icono	Descripción
	Indicates that an attribute value has been defined in a style sheet
	Indicates that an attribute value has been defined in a style sheet with the !important declaration
	Displayed when an attribute value defined in a style sheet for at least one item in a group or a selection of multiple objects is different from the other objects

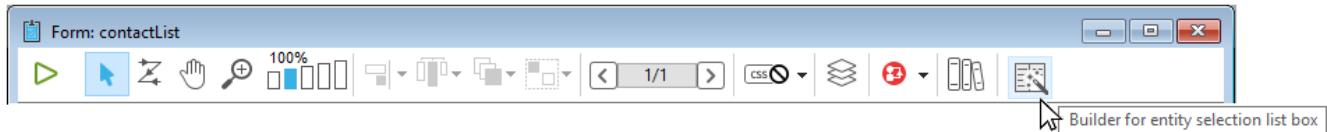
## Creación de list box

Puede crear rápidamente nuevos list box de tipo selección de entidades con el -Generador de list box. El nuevo list box puede ser utilizado inmediatamente o puede ser editado a través del Editor de formularios.

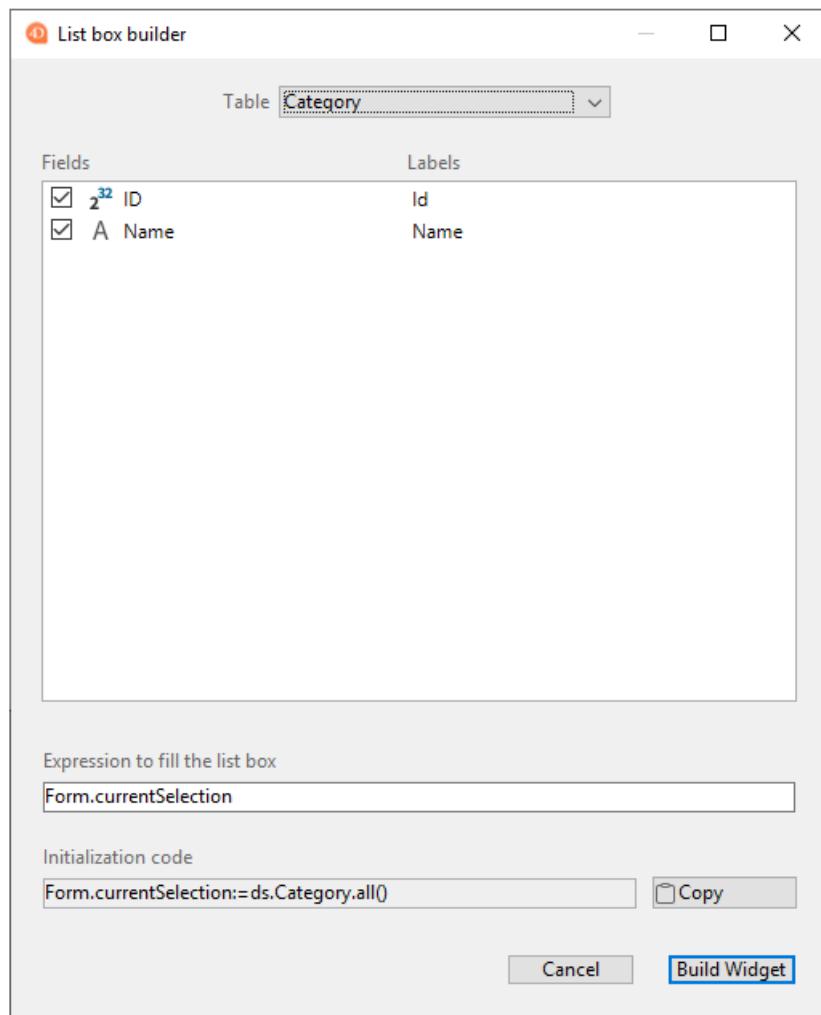
El generador de list box le permite crear y llenar los list box de tipo selección de entidades en unas pocas y sencillas operaciones.

### Utilización del generador de list box

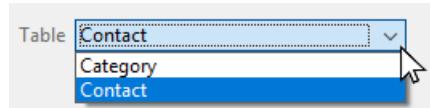
1. En la barra de herramientas del Editor de formularios, haga clic en el ícono del generador de list box:



Se muestra el generador de list box:



2. Seleccione una tabla de la lista desplegable Tabla:



3. Seleccione los campos del list box en el área Campos:

Fields	Labels
<input type="checkbox"/> <sup>32</sup> CategoryID	Category id
<input type="checkbox"/> A Email	Email
<input checked="" type="checkbox"/> A FirstName	First Name
<input type="checkbox"/> A HomeAddress	Home Address
<input type="checkbox"/> A HomeCity	Home City
<input type="checkbox"/> A HomeCountry	Home Country
<input type="checkbox"/> A HomePhone	Home Phone
<input type="checkbox"/> A HomeState	Home State
<input type="checkbox"/> A HomeZip	Home zip
<input type="checkbox"/> <sup>32</sup> ID	Id
<input type="checkbox"/> A JobTitle	Job Title
<input checked="" type="checkbox"/> A LastName	Last Name
<input type="checkbox"/> A MiddleName	Middle Name
<input type="checkbox"/> A MobilePhone	Mobile Phone
<input type="checkbox"/> A Notes	Notes
<input checked="" type="checkbox"/> A Organization	Organization

Por defecto, se seleccionan todos los campos. Puede seleccionar o deselegionar los campos individualmente o utilizar Ctrl+clic (Windows) o Cmd+clic (macOS) para seleccionarlos o deselegionarlos todos a la vez.

Puede cambiar el orden de los campos arrastrándolos y soltándolos.

4. La expresión para llenar las líneas del list box a partir de la selección de la entidad se llena previamente:

Expression to fill the list box

Esta expresión puede modificarse si es necesario.

5. Al hacer clic en el botón **Copiar** se copiará la expresión para cargar todos los registros en la memoria:

Initialization code

```
Form.currentSelection:=ds.Contact.all()
```

Copy

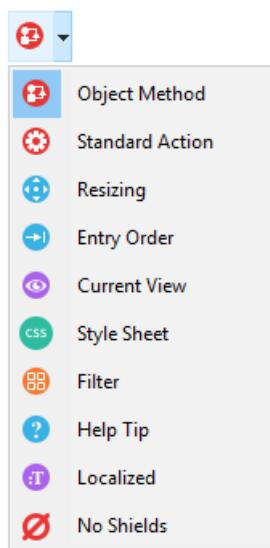
6. Haga clic en el botón Crear un widget para crear el list box.

## Build widget

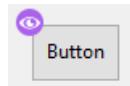
El list box final:

## Marcas

El editor de formularios 4D utiliza marcas para facilitar la visualización de las propiedades de los objetos. Puede encontrarlos en la barra de herramientas del formulario:



El principio de esta función es el siguiente: cada escudo está asociado a una propiedad (por ejemplo, Vistas, que significa que el objeto "está en la vista actual"). Al activar una marca, 4D muestra un pequeño ícono (marca) en la parte superior izquierda de cada objeto del formulario donde se aplica la propiedad.



## Utilizar marcas

Para activar una marca, haga clic en el ícono *Marca* de la barra de herramientas hasta seleccionar la marca deseada. También puede hacer clic en la parte derecha del botón y seleccionar el tipo de marca que desea mostrar directamente en el menú asociado:

Si no quiere mostrar marcas, seleccione Sin marcas en el menú de selección.

La **vista actual** no se puede ocultar.

## Descripciones de marcas

A continuación se describe cada tipo de escudo:

Icono	Nombre	Se muestra...
	Método objeto	Para los objetos con un método objeto asociado
	Acción estándar	Para los objetos con una acción estándar asociada
	Redimensionamiento	Para los objetos con al menos una propiedad de redimensionamiento, indica la combinación de propiedades actuales
	Orden de entrada	En el caso de los objetos editables, indica el número de orden de entrada
	Vista actual	Para todos los objetos de la vista actual
	Hoja de estilo	Para objetos con uno o más valores de atributos reemplazados por una hoja de estilo.
	Filtro	Para los objetos introducibles con un filtro de entrada asociado
	Mensaje de ayuda	Para los objetos con un mensaje de ayuda asociado
	Localizado	Para los objetos cuya etiqueta proviene de una referencia (etiqueta que empieza por ":"). La referencia puede ser de tipo recurso (STR#) o XLIFF
	Sin marcas	No aparecen marcas

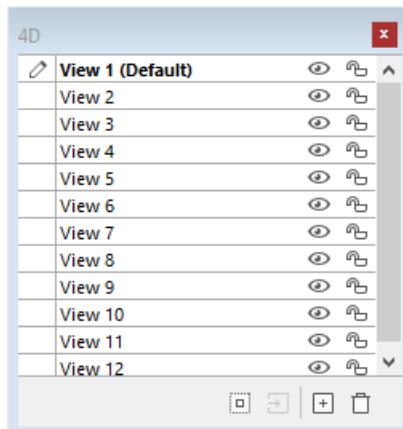
## Vistas

El editor de formularios de 4D le permite crear formularios complejos distribuyendo los objetos formulario entre distintas vistas que pueden ocultarse o mostrarse según sea necesario.

Por ejemplo, puede distribuir los objetos según su tipo (campos, variables, objetos estáticos, etc.). Todo tipo de objeto formulario, incluidos los subformularios y las áreas de plug-in, puede incluirse en las vistas.

No hay límite en el número de vistas por formulario. Puedes crear tantas vistas diferentes como necesite. Además, cada vista puede mostrarse, ocultarse y/o bloquearse.

La gestión de las vistas se realiza a través de la paleta de vistas.



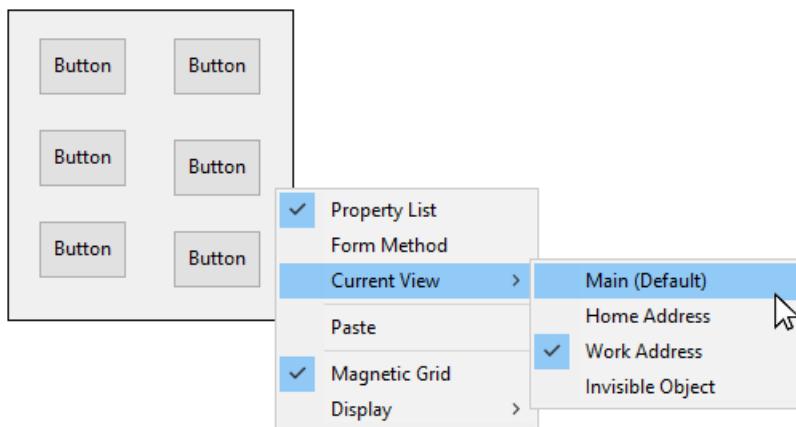
## Acceder a la paleta de vistas

Hay tres formas de acceder a la paleta de vistas:

- Barra de herramientas: haga clic en el ícono Vistas de la barra de herramientas del Editor de formularios. (Este ícono aparece en gris cuando al menos un objeto pertenece a una vista distinta de la vista por defecto.)

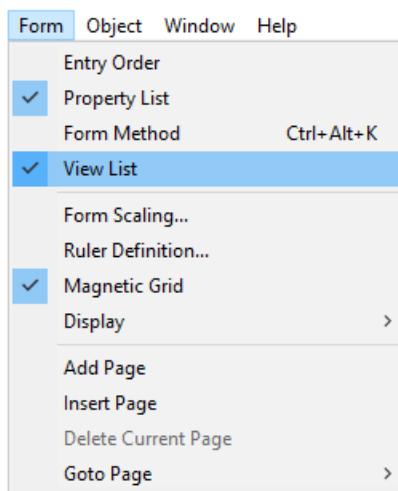


- Menú contextual (formulario u objeto): haga clic derecho en cualquier lugar del editor de formularios o de un objeto, y seleccione Vista actual



La vista actual se indica con una marca de verificación (por ejemplo, "Dirección de trabajo" en la imagen superior)

- Menú Formulario: haga clic en el menú Formulario y seleccione Mostrar la lista



## Antes de comenzar

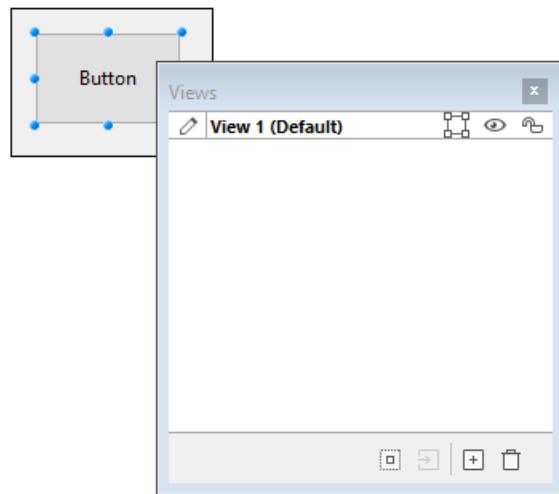
Aquí hay algunas cosas importantes que hay que saber antes de empezar a trabajar con vistas:

- Contexto de uso: las vistas son una herramienta puramente gráfica que sólo se puede utilizar en el Editor de formularios; no se puede acceder a las vistas por programación ni en el modo Aplicación.
- Vistas y páginas: Los objetos de una misma vista pueden pertenecer a diferentes páginas del formulario; sólo se pueden mostrar los objetos de la página actual (y de la página 0 si es visible), independientemente de la configuración de las vistas.
- Vistas y niveles: las vistas son independientes de los niveles de los objetos; no existe una jerarquía de visualización entre las diferentes vistas.
- Vistas y grupos: sólo se pueden agrupar los objetos que pertenecen a la vista actual.
- Vistas actuales y por defecto: la vista por defecto es la primera vista de un formulario y no se puede eliminar; la vista actual es la que se está editando y el nombre se muestra en negrita.

## Gestión de vistas

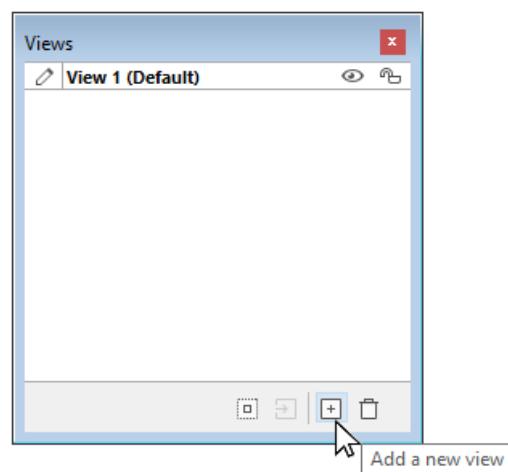
### Crear vistas

Todo objeto creado en un formulario se coloca en la primera vista ("Vista 1") del formulario. La primera vista es siempre la vista por defecto, indicada por (por defecto) después del nombre. El nombre de la vista puede cambiarse (ver [Renombrar vistas](#)), sin embargo sigue siendo la vista por defecto.

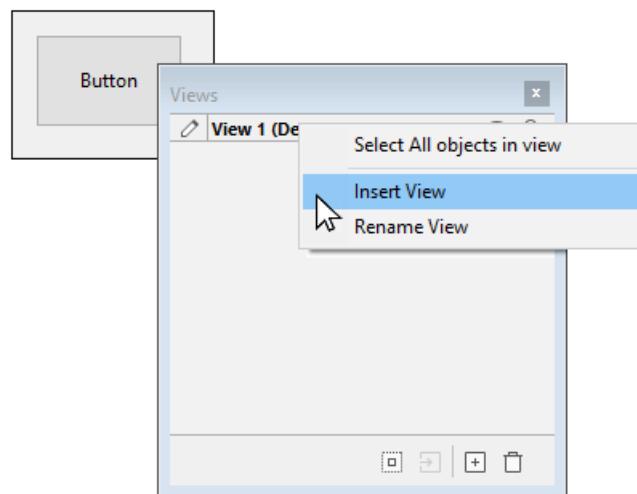


Hay dos maneras de añadir vistas adicionales:

- Haga clic en el botón Añadir una nueva vista en la parte inferior de la paleta Vista:



- Haga clic con el botón derecho en una vista existente y seleccione Insertar vista:



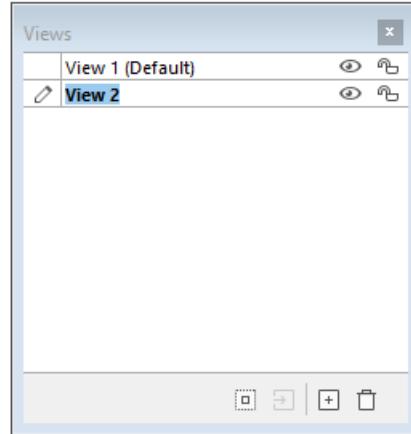
No hay límite en el número de vistas.

#### Renombrar vistas

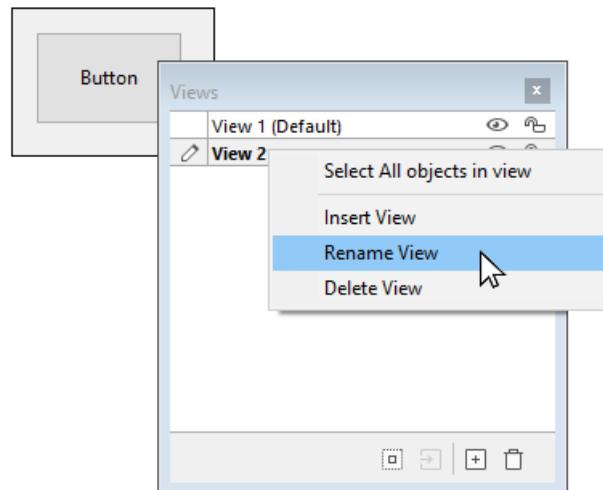
Por defecto las vistas se nombran como "Vista" + el número de la vista, sin embargo puede cambiar estos nombres para mejorar la legibilidad y adaptarse mejor a sus necesidades.

Para cambiar el nombre de una vista, puede utilizar:

- Hacer doble clic directamente en el nombre de la vista (la vista seleccionada en este caso). El nombre se convierte entonces en editable:



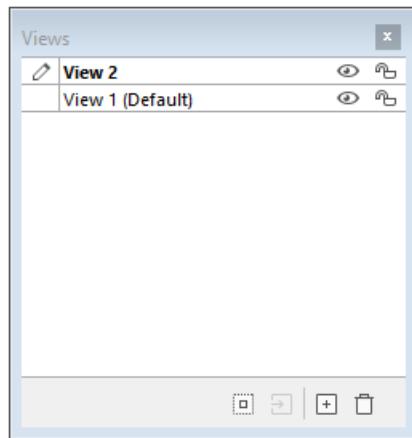
- Haga clic derecho en el nombre de la vista. El nombre se convierte entonces en editable:



## Reordenar las vistas

Puede cambiar el orden de visualización de las vistas haciendo arrastrar y soltar dentro de la paleta de vistas.

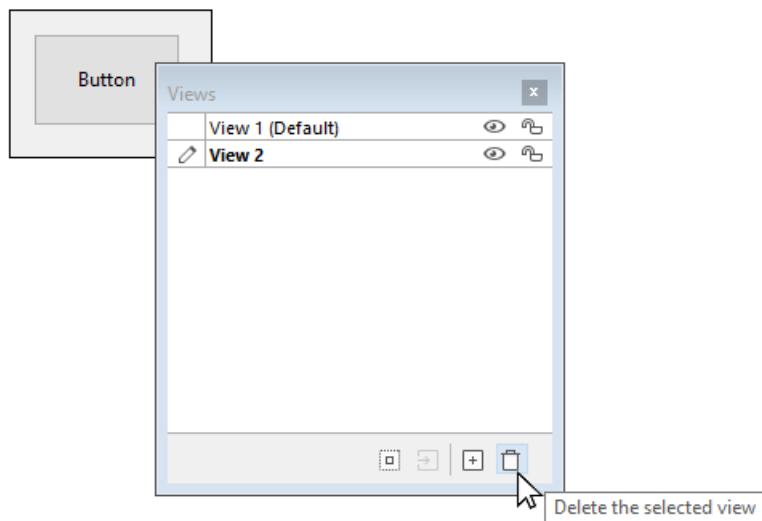
Tenga en cuenta que la vista por defecto no cambia:



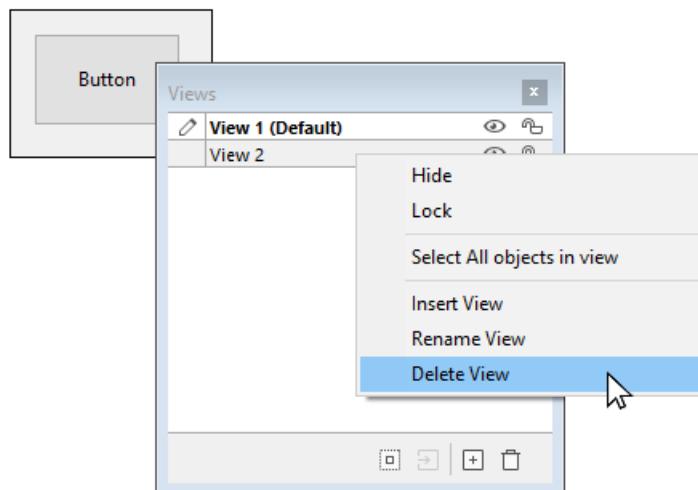
## Eliminar vistas

Para cambiar el nombre de una vista, puede utilizar:

- Haga clic en el botón Eliminar la vista seleccionada en la parte inferior de la paleta Vista:



- Haga clic derecho en el nombre de la vista y seleccione Eliminar la vista:



La **vista actual** no se puede bloquear.

## Utilización de las vistas

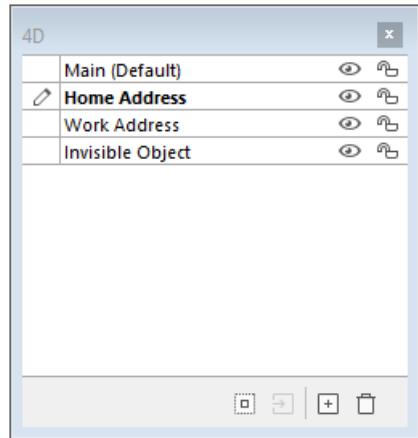
Una vez creadas las vistas, puede utilizar la paleta de vistas para:

- Añadir un objeto a las vistas,
- Mover los objetos de una vista a otra,
- Seleccionar todos los objetos de la misma vista con un solo clic,
- Mostrar u ocultar objetos para cada vista,
- Bloquear los objetos de una vista.

### Añadir los objetos a las vistas

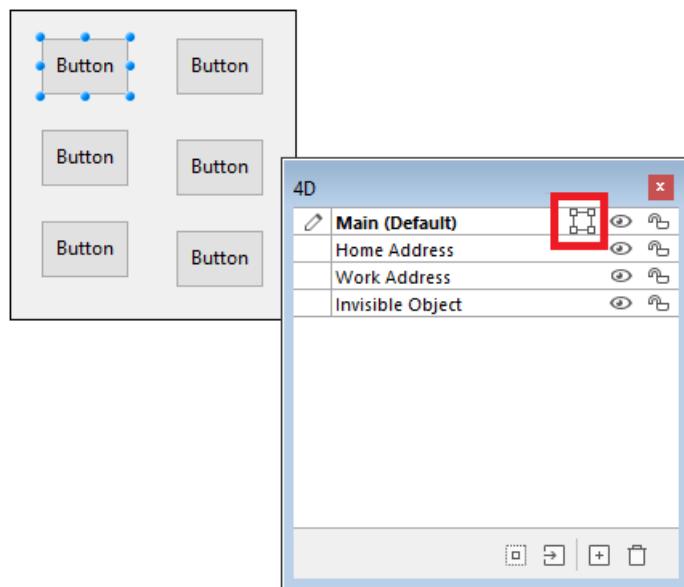
Un objeto sólo puede pertenecer a una única vista.

Para crear un objeto en otra vista, basta con seleccionar la vista en la paleta de vistas (antes de crear el objeto) haciendo clic en su nombre (se muestra un ícono de edición para la **Vista actual** y el nombre aparece en negrita):



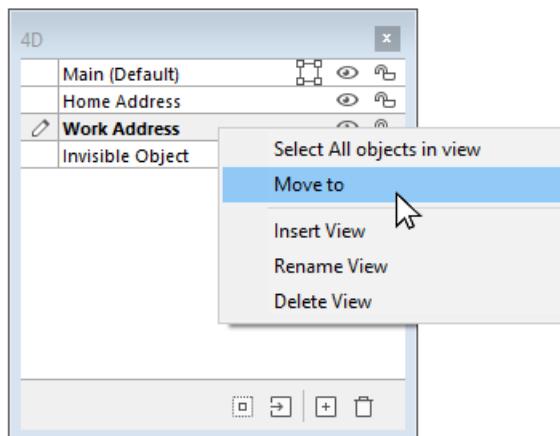
## Mover objetos entre vistas

También es posible mover uno o más objetos de una vista a otra. En el formulario, seleccione el o los objetos cuya vista desea modificar. La lista de vistas indica, utilizando un símbolo, la vista a la que pertenece la selección:



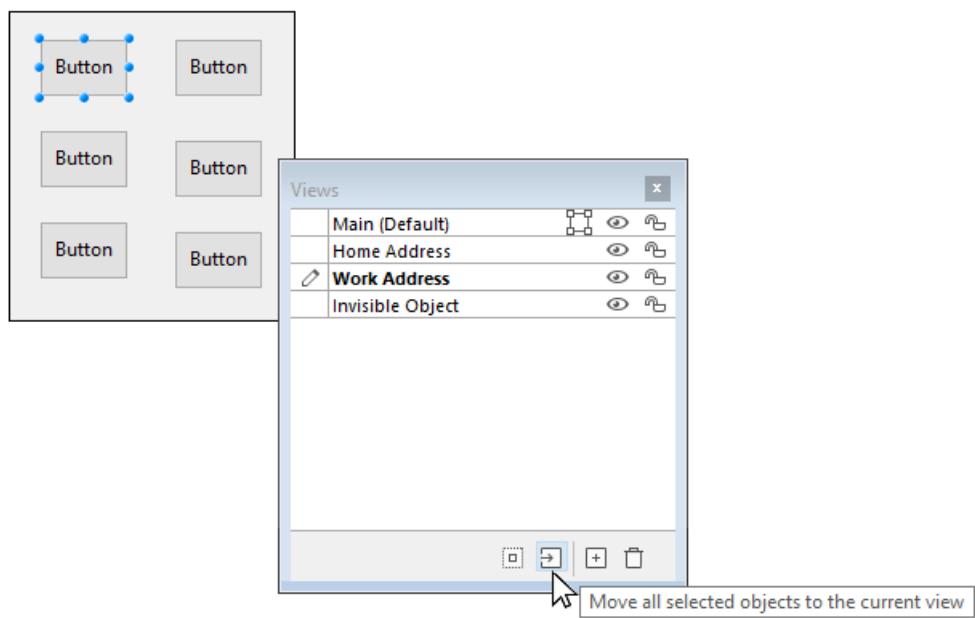
La selección puede contener varios objetos pertenecientes a diferentes vistas.

Simplemente seleccione la vista de destino, haga clic derecho y seleccione Mover a:

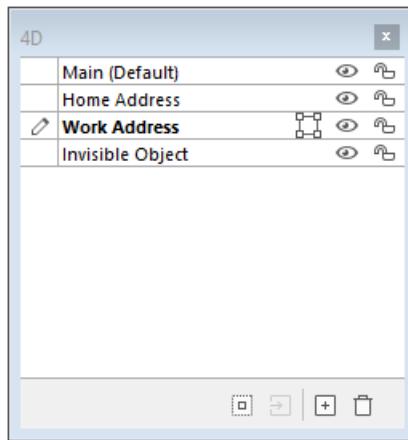


O

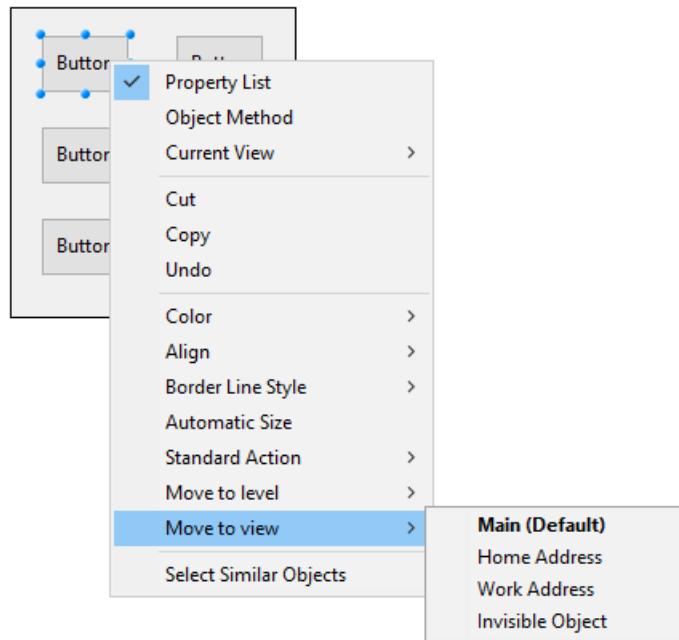
Seleccione la vista de destino de la selección y haga clic en el botón Mover a de la parte inferior de la paleta de vistas:



La selección se coloca entonces en la nueva vista:



También puede mover un objeto a otra vista a través del menú contextual del objeto. Haga clic derecho en el objeto, seleccione Mover a la vista y seleccione una vista en la lista de vistas disponibles:

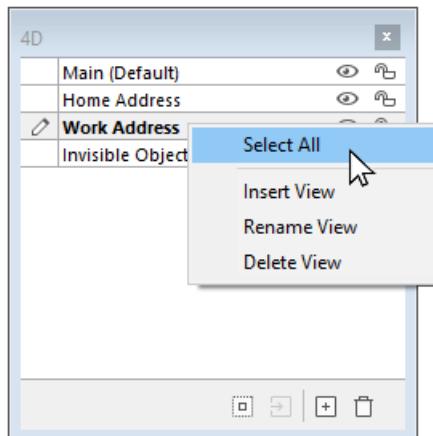


La **vista actual** se muestra en negrita.

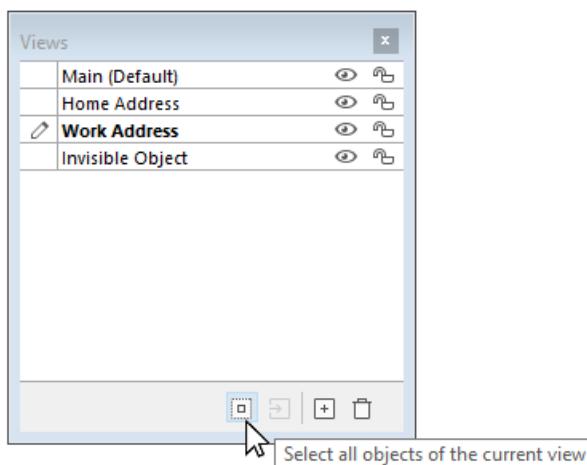
Seleccionar todos los objetos de una vista

Puede seleccionar todos los objetos que pertenecen a la misma vista en la página actual del formulario. Esta función es útil para aplicar cambios globales a un conjunto de objetos.

Para ello, haga clic derecho en la vista en la que desea seleccionar todos los objetos, haga clic en **Seleccionar todo**:



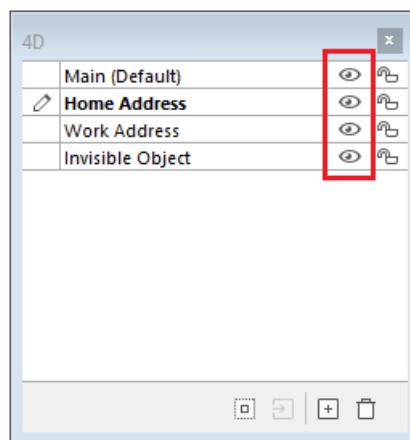
También puede utilizar el botón situado en la parte inferior de la paleta de vistas:



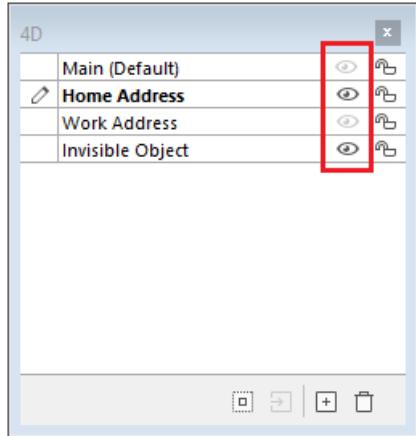
#### Mostrar u ocultar los objetos de una vista

Puede mostrar u ocultar objetos pertenecientes a una vista en cualquier momento en la página actual del formulario. De este modo, podrá centrarse en determinados objetos al editar el formulario, por ejemplo.

Por defecto, se muestran todas las vistas, como indica el icono *Mostrar/Ocultar*:



Para ocultar una vista, haga clic en el icono *Mostrar/Ocultar*. Entonces se atenua y los objetos de la vista correspondiente dejan de mostrarse en el formulario:



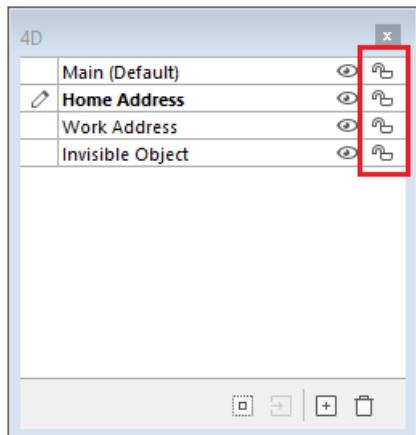
La **vista actual** no se puede ocultar.

Para mostrar una vista que está oculta, simplemente selecciónela o haga clic en el icono *Mostrar/Ocultar de esa vista*.

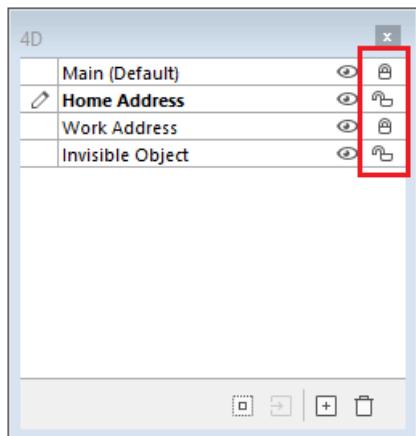
#### Bloquear los objetos de una vista

Puede bloquear los objetos de una vista. Esto impide que se seleccionen, modifiquen o eliminen del formulario. Una vez bloqueado, un objeto no puede seleccionarse mediante un clic, un rectángulo o el comando *Seleccionar objetos* similares del menú contextual. Esta función es útil para evitar errores de manipulación.

Por defecto, todas las vistas están desbloqueadas, como lo indica el icono *Bloquear/Desbloquear* que aparece junto a cada vista:



Para bloquear los objetos de una vista, haga clic en el icono *Bloquear/Desbloquear*. El candado está cerrado, lo que significa que la vista está bloqueada:



La **vista actual** no se puede bloquear.

Para desbloquear una vista que está bloqueada, basta con seleccionarla o hacer clic en el icono *Bloquear/Desbloquear*

de esa vista.

## Zoom

Puede hacer zoom en el formulario actual. Pase al modo "Zoom" haciendo clic en el icono de la lupa o haciendo clic directamente en la barra de porcentaje deseada (50%, 100%, 200%, 400% y 800%):



- Al hacer clic en la lupa, el cursor se convierte en una lupa. A continuación, puede hacer clic en el formulario para aumentar la visualización o mantener presionada la tecla Mayús y hacer clic para reducir el porcentaje de visualización.
- Al hacer clic en una barra de porcentaje, la visualización se modifica inmediatamente.

En el modo Zoom, todas las funciones del editor de formularios siguen estando disponibles(\*) .

(\*) Por razones técnicas, no es posible seleccionar los elementos del list box (encabezados, columnas, pies de página) cuando el editor de formularios está en modo Zoom.

# Macros del editor de formularios

El editor de formularios 4D soporta macros. Una macro es un conjunto de instrucciones que permiten realizar una acción o una secuencia de acciones. Cuando se llama, la macro ejecutará sus instrucciones y realiza automáticamente la(s) acción(es).

Por ejemplo, si tiene un informe recurrente con un formato específico (por ejemplo, cierto texto debe aparecer en rojo y cierto texto debe aparecer en verde), puede crear una macro para definir automáticamente el color. Puede crear macros para el editor de formularios 4D que pueden:

- Crear y ejecutar código 4D
- Mostrar las cajas de diálogo
- Seleccione los objetos de formulario
- Añadir / eliminar / modificar los formularios, los objetos de formulario así como sus propiedades
- Modificar los archivos del proyecto (actualizar, eliminar)

El código de las macros soporta [funciones de clase](#) y las [propiedades de objeto de formulario en JSON](#) para permitir definir toda funcionalidad personalizada en el editor de formularios.

Las macros pueden definirse para el proyecto local o para componentes dentro del proyecto. Por lo general, se crea una macro y se instala dentro de los componentes que se utilizan para el desarrollo.

Cuando se llama, una macro anula todo comportamiento especificado previamente.

## Ejemplo práctico

En este breve ejemplo, verá cómo crear y llamar a una macro que añade un botón de alerta "Hello World" en la esquina superior izquierda de su formulario.

1. En un archivo `formMacros.json` dentro de la carpeta `Sources` de su proyecto, escriba:

```
{  
  "macros": {  
    "Add Hello World button": {  
      "class": "AddButton"  
    }  
  }  
}
```

2. Cree una clase 4D llamada `AddButton`.
3. En la clase `AddButton`, escriba la siguiente función:

```

Function onInvoke($editor : Object)->$result : Object

    var $btnHello : Object

    // Crea un botón "Hello"
    $btnHello:=New object("type"; "button"; \
    "text"; "Hello World!"; \
    "method"; New object("source"; "ALERT(\"Hello World!\")"); \
    "events"; New collection("onClick"); \
    "width"; 120; \
    "height"; 20; \
    "top"; 0; \
    "left"; 0)

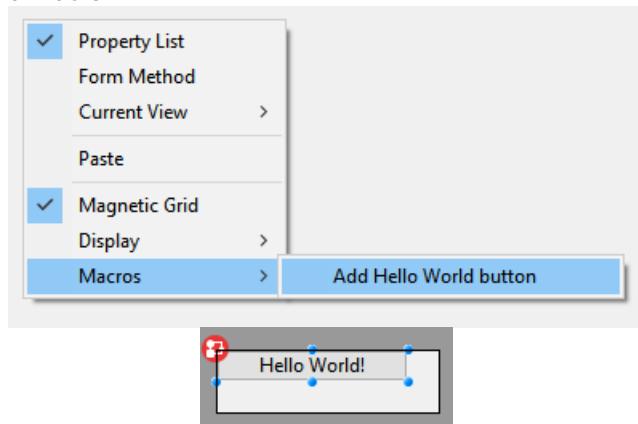
    // Añadir un botón en la página actual
    $editor.editor.currentPage.objects.btnHello:=$btnHello

    // Seleccionar el nuevo botón en el editor de formularios
    $editor.editor.currentSelection.clear() //deseleccionar elementos
    $editor.editor.currentSelection.push("btnHello")

    //Notificar la modificación al editor de formularios 4D
    $result:=New object("currentSelection"; $editor.editor.currentSelection;\n
        "currentPage"; $editor.editor.currentPage)

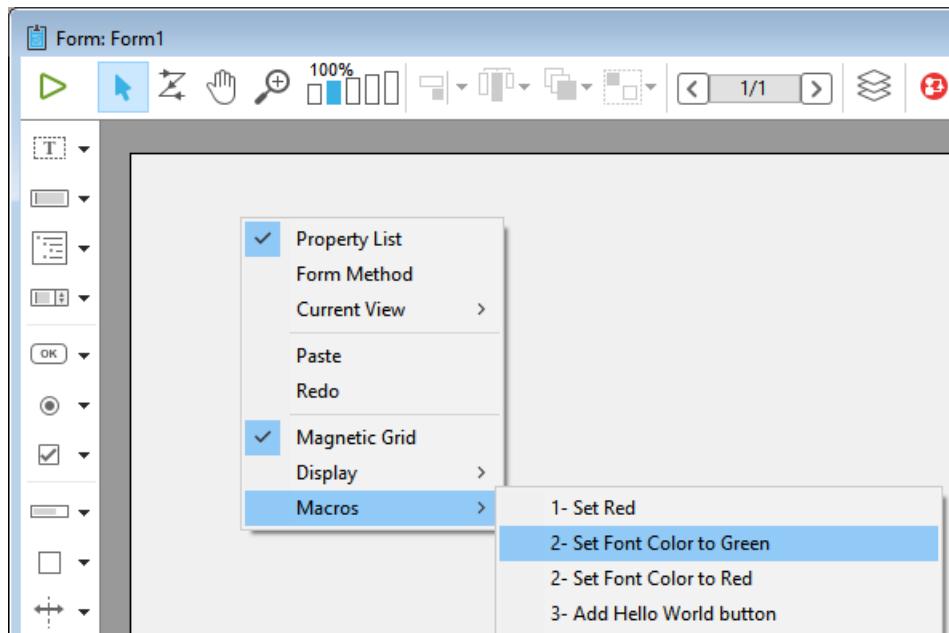
```

A continuación, puede llamar a la macro:



## Llamar a las macros en el editor de formularios

Cuando las macros están definidas en su proyecto 4D, puede llamar una macro utilizando el menú contextual del editor de formularios:



Este menú se crea sobre el [archivo de definición de macros](#) `formMacros.json`. Los elementos de la macro se clasifican en orden alfabético.

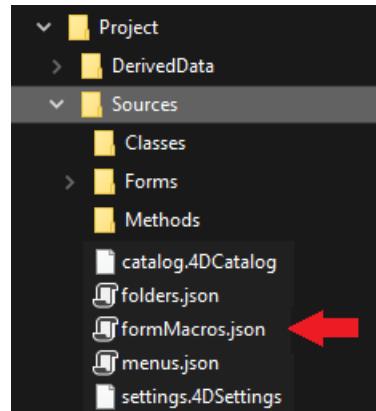
Este menú puede ser llamado en un área vacía o en una selección en el formulario. El objeto seleccionado se pasa a `$editor.currentSelection` o `$editor.target` en la función `onInvoke` de la macro.

Una sola macro puede ejecutar varias operaciones. Si se selecciona, la función Deshacer del editor de formularios puede utilizarse para revertir las operaciones de las macros de forma global.

## Ubicación del archivo de macro

Todas las macros del editor de formularios 4D se definen en un único archivo JSON por proyecto o componente: `FormMacros.json`.

Este archivo debe estar ubicado en la carpeta Project > Sources local o del componente:



## Declaración de macros

La estructura del archivo `formMacros.json` es la siguiente:

```
{
  "macros": {
    <macroName>: {
      "class": <className>,
      <customProperty> : <value>
    }
  }
}
```

Esta es la descripción del contenido del archivo JSON:

Atributo			Tipo	Descripción
macros			objeto	lista de macros definidas
	<macroName>		objeto	definición de la macro
		class	cadena	nombre de clase de la macro
		<customProperty>	any	(opcional) valor personalizado a recuperar en el constructor

Las propiedades personalizadas, cuando se utilizan, se pasan a la función [constructor](#) de la macro.

## Ejemplo

```
{
  "macros": {
    "Open Macros file": {
      "class": "OpenMacro"
    },
    "Align to Right on Target Object": {
      "class": "AlignOnTarget",
      "myParam": "right"
    },
    "Align to Left on Target Object": {
      "class": "AlignOnTarget",
      "myParam": "left"
    }
  }
}
```

## Instanciar las macros en 4D

Cada macro que quiera instanciar en su proyecto o componente debe ser declarada como una [clase 4D](#).

El nombre de la clase debe coincidir con el nombre definido mediante el atributo [class](#) del archivo [formMacros.json](#).

Las macros se instancian al iniciar la aplicación. Por lo tanto, si se modifica la estructura de la clase de macro (añadir una función, modificar un parámetro... o el [constructor](#) , tendrá que reiniciar la aplicación para aplicar los cambios.

## Funciones macro

Cada clase de macro puede contener un [Class constructor](#) y dos funciones: [onInvoke\(\)](#) y [onError\(\)](#) .

### Class constructor

[Class constructor\(\\$macro : Object\)](#)

Parámetros	Tipo	Descripción
\$macro	Objeto	Objeto de declaración de macros (en el archivo <code>formMacros.json</code> )

Las macros se instancian utilizando una función [class constructor](#), si existe.

El class constructor se llama una vez durante la instanciación de clase, que se produce al inicio de la aplicación.

Las propiedades personalizadas añadidas a la [declaración macro](#) se devuelven en el parámetro de la función class constructor.

## Ejemplo

En el archivo `formMacros.json`:

```
{
  "macros": {
    "Align to Left on Target Object": {
      "class": "AlignOnTarget",
      "myParam": "left"
    }
  }
}
```

Puede escribir:

```
// Class "AlignOnTarget"
Class constructor($macro : Object)
    This.myParameter:=$macro.myParam //left
    ...
}
```

onInvoke()

onInvoke(\$editor : Object) -> \$result : Object

Parámetros	Tipo	Descripción
\$editor	Objeto	Objeto Form Editor Macro Proxy que contiene las propiedades del formulario
\$result	Objeto	Objeto Form Editor Macro Proxy que devuelve las propiedades modificadas por la macro (opcional)

La función `onInvoke` se ejecuta automáticamente cada vez que se llama a la macro.

Cuando la función es llamada, recibe en la propiedad `$editor.editor` una copia de todos los elementos del formulario con sus valores actuales. Luego puede ejecutar cualquier operación en estas propiedades.

Una vez completadas las operaciones, si la macro resulta en la modificación, adición o eliminación de objetos, puede pasar las propiedades editadas resultantes en `$result`. El procesador de macros analizará las propiedades devueltas y aplicará las operaciones necesarias en el formulario. Obviamente, cuanto menos propiedades devuelva, menos tiempo requerirá el procesamiento.

Estas son las propiedades devueltas en el parámetro `$editor`:

Propiedad	Tipo	Descripción
\$editor.editor.form	Objeto	Formulario completo
\$editor.editor.file	File	File object of the form file
\$editor.editor.name	Cadena	Nombre del formulario
\$editor.editor.table	number	Número de tabla del formulario, 0 para el formulario proyecto
\$editor.editor.currentPageNumber	number	The number of the current page
\$editor.editor.currentPage	Objeto	The current page, containing all the form objects and the entry order of the page
\$editor.editor.currentSelection	Collection	Colección de nombres de objetos seleccionados
\$editor.editor.formProperties	Objeto	Properties of the current form
\$editor.editor.target	cadena	Name of the object under the mouse when clicked on a macro

Here are the properties that you can pass in the `$result` object if you want the macro processor to execute a modification. Todas las propiedades son opcionales:

Propiedad	Tipo	Descripción
currentPage	Objeto	currentPage including objects modified by the macro, if any
currentSelection	Collection	currentSelection si es modificada por la macro
formProperties	Objeto	formProperties si es modificado por la macro
editor.groups	Objeto	información de grupo, si los grupos son modificados por la macro
editor.views	Objeto	ver información, si las vistas son modificadas por la macro
editor.activeView	Cadena	Nombres de vistas activos

For example, if objects of the current page and groups have been modified, you can write:

```
$result:=New object("currentPage"; $editor.editor.currentPage ; \
"editor"; New object("groups"; $editor.editor.form.editor.groups))
```

atributo `method`

When handling the `method` attribute of form objects, you can define the attribute value in two ways in macros:

- Using a [string containing the method file name/path](#).
- Utilizando un objeto con la siguiente estructura:

Propiedad	Tipo	Descripción
source	Cadena	Código del método

4D will create a file using the object name in the "objectMethods" folder with the content of `source` attribute. Esta función solo está disponible para el código macro.

Propiedad `$4dId` en `currentPage.objects`

La propiedad `$4dId` define un ID único para cada objeto de la página actual. Esta clave es utilizada por el procesador de macros para controlar los cambios en `$result.currentPage`:

- si la llave `$4dId` falta tanto en el formulario y en un objeto en `$result`, el objeto se crea.
- si la llave `$4dId` existe en el formulario pero falta en `$result`, el objeto se elimina.

- si la llave `$4dId` existe tanto en el formulario y en un objeto en `$result`, el objeto se modifica.

## Ejemplo

Quiere definir una función macro que aplique el color rojo y el estilo de letra itálica a cualquier objeto seleccionado.

```
Function onInvoke($editor : Object)->$result : Object
  var $name : Text

  If ($editor.editor.currentSelection.length>0)
    // Define el contorno en rojo y el estilo en itálica para cada objeto seleccionado
    For each ($name; $editor.editor.currentSelection)
      $editor.editor.currentPage.objects[$name].stroke:="red"
      $editor.editor.currentPage.objects[$name].fontStyle:="italic"

    End for each

  Else
    ALERT("Please select a form object.")
  End if

  // Notificar la modificación a 4D
  $result:=New object("currentPage"; $editor.editor.currentPage)
```

## onError()

onError(\$editor : Object; \$resultMacro : Object ; \$error : Collection)

Parámetros		Tipo	Descripción
\$editor		Objeto	Objeto enviado a <code>onInvoke</code>
\$resultMacro		Objeto	Objeto devuelto por <code>onInvoke</code>
\$error		Collection	Pila de errores
	[]errorCode	Número	Código de error
	[]message	Texto	Descripción del error
	[]componentSignature	Texto	Firma del componente interno

La función `onError` se ejecuta cuando el procesador de macros encuentra un error.

Cuando se ejecuta una macro, si 4D encuentra un error que impide la cancelación de la macro, no la ejecuta. Es el caso, por ejemplo, de que la ejecución de una macro resulte en:

- borrar o modificar un script cuyo archivo es de sólo lectura.
- crear dos objetos con el mismo ID interno.

## Ejemplo

En la definición de una clase macro, se puede escribir el siguiente código de error genérico:

```
Function onError($editor : Object; $resultMacro : Object; $error : Collection)
    var $obj : Object
    var $txt : Text
    $txt:=""

    For each ($obj; $error)
        $txt:=$txt+$obj.message+"\n"
    End for each

    ALERT($txt)
```

# Librerías de objetos

Puede utilizar librerías de objetos en sus formularios. Una librería de objetos ofrece una colección de objetos preconfigurados que pueden ser utilizados en sus formularios mediante un simple copiar y pegar o arrastrar y soltar.

4D propone dos tipos de librerías de objetos:

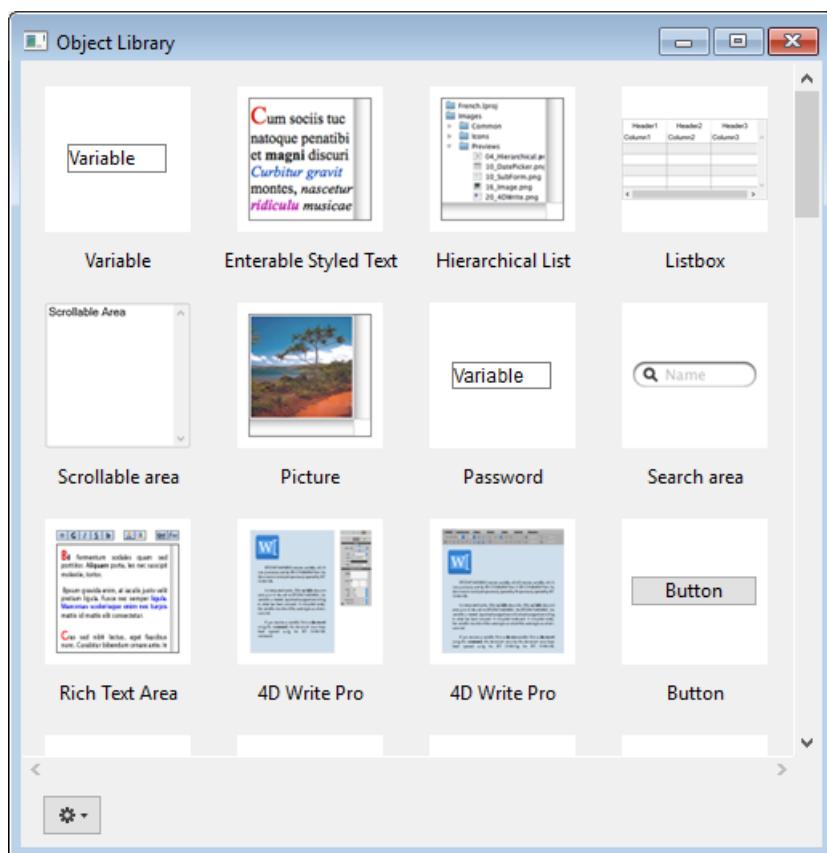
- una librería de objetos estándar y preconfigurada, disponible en todos sus proyectos.
- librerías de objetos personalizadas, que puede utilizar para almacenar sus objetos formularios favoritos o formularios proyecto completos.

## Utilización de la librería de objetos estándar

La librería de objetos estándar está disponible en el editor de formularios: haga clic en el último botón de la barra de herramientas:

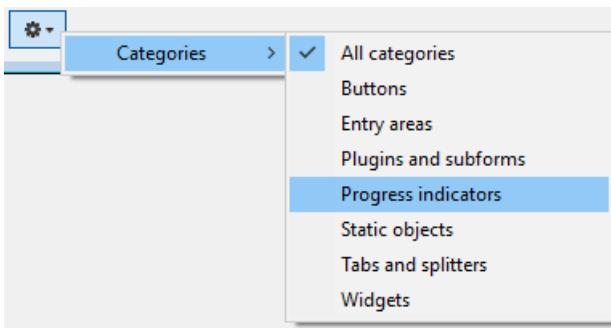


La librería se muestra en una ventana aparte:



La ventana tiene las siguientes características principales:

- Área de vista previa con mensajes de ayuda: el área central muestra una vista previa de cada objeto. Puede pasar el ratón por encima de un objeto para obtener información sobre el mismo en un mensaje de ayuda.
- Puede filtrar el contenido de la ventana utilizando el menú Categorías:



- Para utilizar un objeto de la librería en su formulario, puede:
  - hacer clic derecho en un objeto y seleccionar Copiar en el menú contextual
  - o arrastrar y soltar el objeto desde la librería El objeto se añade al formulario.

Esta librería es de sólo lectura. Si desea editar objetos por defecto o crear su propia librería de objetos preconfigurados o formularios proyecto, deberá crear una librería de objetos personalizada (ver abajo).

Todos los objetos propuestos en la librería de objetos estándar se describen en [esta sección en doc.4d.com](#).

## Crear y utilizar librerías de objetos personalizadas

Puede crear y utilizar librerías de objetos personalizadas en 4D. Una librería de objetos personalizada es un proyecto 4D donde puede almacenar sus objetos favoritos (botones, textos, imágenes, etc.) A continuación, puede reutilizar estos objetos en diferentes formularios y proyectos. A continuación, puede reutilizar estos objetos en diferentes formularios y proyectos.

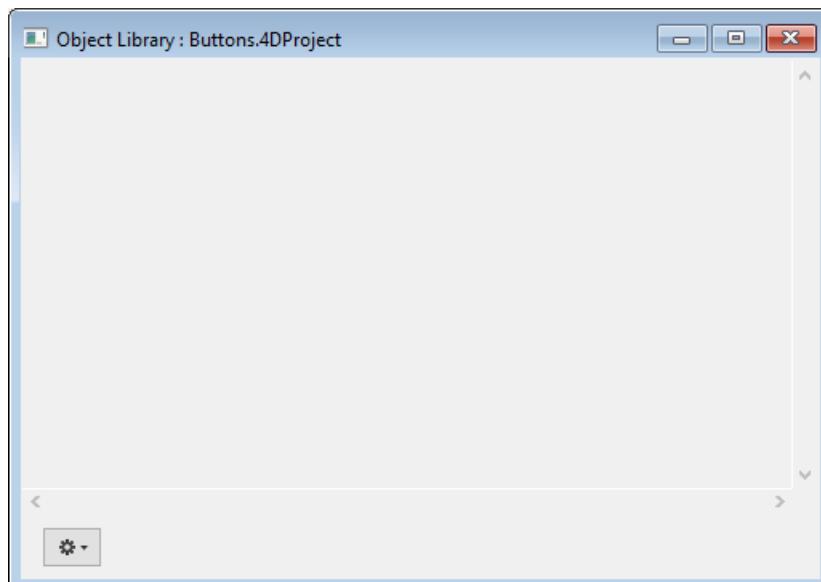
Los objetos se almacenan con todas sus propiedades, incluidos sus métodos. Las librerías se arman y utilizan mediante simples operaciones de arrastrar y soltar o copiar y pegar.

Mediante el uso de librerías, se pueden construir fondos de objetos de formulario agrupados por familias gráficas, por funcionalidades, etc.

### Crear una librería de objetos

Para crear una librería de objetos, seleccione Nuevo>Librería de objetos... en el menú Archivo o en la barra de herramientas de 4D. Aparece una caja de diálogo estándar para guardar el archivo, que le permite elegir el nombre y la ubicación de la librería de objetos.

Una vez validada la caja de diálogo, 4D crea una nueva librería de objetos en su disco y muestra su ventana (vacía por defecto).



Puede crear tantas librerías como desee por proyecto. Una librería creada y construida en macOS puede utilizarse en Windows y viceversa.

## Abrir una librería de objetos

Una determinada librería de objetos sólo puede ser abierta por un proyecto a la vez. Sin embargo, se pueden abrir varias librerías diferentes en el mismo proyecto.

Para abrir una librería de objetos personalizada, seleccione el comando Abrir>Librería de objetos... en el menú Archivo o en la barra de herramientas de 4D. Aparece una caja de diálogo estándar para abrir archivos, que le permite seleccionar la librería de objetos que desea abrir. Puede seleccionar los siguientes tipos de archivos:

- .4dproject
- .4dz

De hecho, las librerías de objetos personalizadas son proyectos 4D clásicos. Sólo se exponen las siguientes partes de un proyecto cuando se abre como librería:

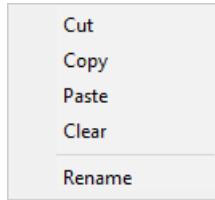
- formularios proyecto
- páginas formulario 1

## Crear una librería de objetos

Los objetos se colocan en una librería de objetos mediante una operación de arrastrar y soltar o de cortar, copiar y pegar. Pueden provenir de un formulario o de otra librería de objetos (incluyendo la [librería estándar](#)). No se conserva ningún enlace con el objeto original: si el original se modifica, el objeto copiado no se ve afectado.

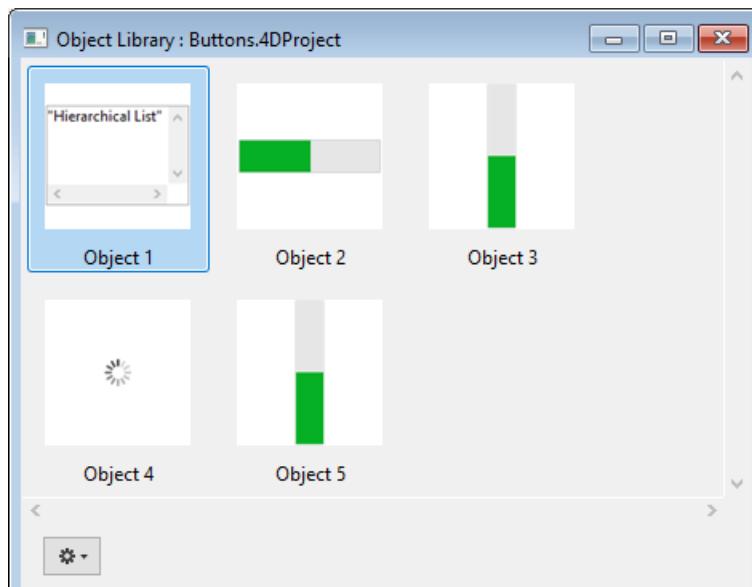
Para poder arrastrar y soltar objetos de los formularios a las librerías de objetos, debe asegurarse de seleccionar la opción Iniciar arrastrar y soltar en las Preferencias de 4D.

Las principales operaciones están disponibles en el menú contextual o en el menú de opciones de la ventana:



- Cortar o Copiar al portapapeles
- Pegar un objeto del tablero portapapeles
- Borrar - elimina el objeto de la librería
- Renombrar - aparece una caja de diálogo que permite cambiar el nombre del elemento. Tenga en cuenta que los nombres de los objetos deben ser únicos en una librería.

Puede colocar objetos individuales (incluidos los subformularios) o conjuntos de objetos en una librería de objetos. Cada objeto o conjunto se agrupa en un solo elemento:



Una librería de objetos puede contener hasta 32.000 elementos.

Los objetos se copian con todas sus propiedades, tanto gráficas como funcionales, incluidos sus métodos. Estas propiedades se mantienen en su totalidad cuando el elemento se copia en un formulario o en otra librería.

### Objetos dependientes

El uso de copiar y pegar o arrastrar y soltar con ciertos objetos de librería también hace que se copien sus objetos dependientes. Por ejemplo, si se copia un botón, se copiará también el método del objeto que se pueda adjuntar. Estos objetos dependientes no se pueden copiar ni arrastrar y soltar directamente.

A continuación se muestra una lista de objetos dependientes que se pegarán en la librería al mismo tiempo que el objeto principal que los utiliza (cuando corresponda):

- Listas
- Formatos/Filtros
- Imágenes
- Mensajes de ayuda (asociados a un campo)
- Métodos objeto

# Hojas de estilo

Una hoja de estilo agrupa una combinación de atributos de objetos formulario, desde los atributos de texto hasta casi todos los atributos de objeto disponibles.

Además de armonizar la interfaz de sus aplicaciones, las hojas de estilo ofrecen tres grandes ventajas:

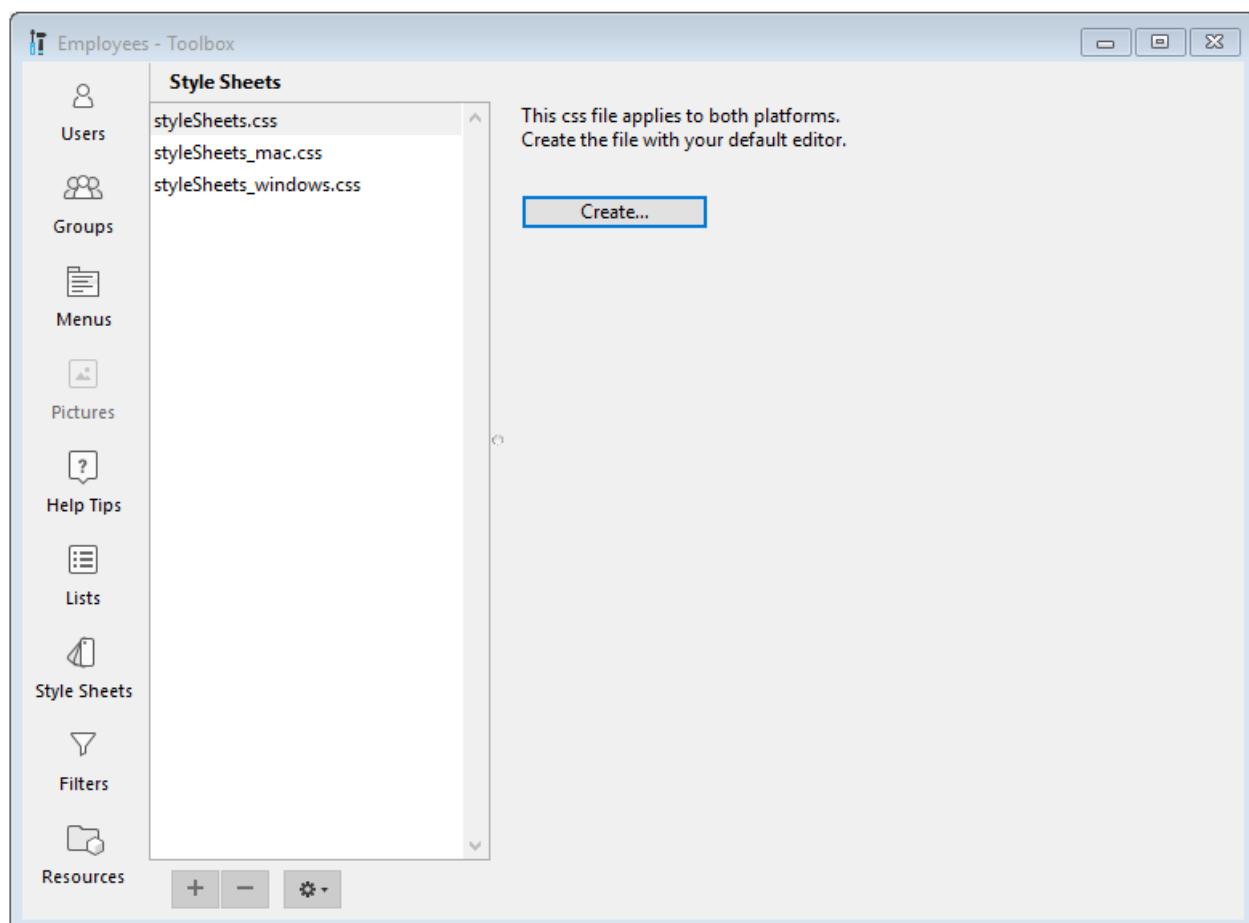
- Permite ahorrar tiempo durante el desarrollo: para cada objeto tiene un grupo específico de parámetros dentro de una sola operación.
- Facilita el mantenimiento: las hojas de estilo modifican la apariencia de todos los objetos que las utilicen, por lo que cambiar el tamaño de la fuente en una hoja de estilo cambiará el tamaño de la fuente para todos los objetos que utilicen esta misma hoja de estilo.
- Control del desarrollo multiplataforma: las hojas de estilo se pueden aplicar a las plataformas macOS y Windows, sólo a macOS o sólo a Windows. Cuando se aplica una hoja de estilo, 4D utiliza automáticamente la hoja de estilo apropiada.

## Creación o modificación de hojas de estilo

Puede crear hojas de estilo utilizando su editor de texto preferido y guardando el archivo con extensión ".css" en la carpeta "/SOURCES" del proyecto.

La caja de herramientas de 4D ofrece una página Hojas de estilo como opción de acceso directo para crear y editar una de las tres hojas de estilo con nombre específicas de la plataforma.

1. Abra la página Estilos eligiendo la Caja de herramientas > Styles del menú Diseño o haga clic en el ícono Caja de herramientas de la barra de herramientas del editor de formularios.



2. Seleccione el tipo de hoja de estilo que desea crear y haga clic en el botón Crear o Editar:

Create...

3. La hoja de estilo se abrirá en su editor de texto predeterminado.

# Archivos hojas de estilo

4D acepta tres archivos específicos de hojas de estilo:

Hoja de estilo	Plataforma
styleSheets.css	Hoja de estilo global por defecto para macOS y Windows
styleSheets_mac.css	Para definir los estilos de atributos específicos de macOS únicamente
styleSheets_windows.css	Para definir los estilos de atributos específicos para Windows únicamente

Estos archivos se almacenan en la carpeta "/SOURCES" del proyecto. También se puede acceder directamente a través del [CSS Preview](#) en la barra de herramientas del editor de formularios.

## Arquitectura de las hojas de estilo

Aunque adaptadas para satisfacer las necesidades específicas de los formularios 4D, las hojas de estilo para proyectos aplicación suelen seguir la sintaxis y la gramática CSS2.

Cada regla de estilo de una hoja de estilo contiene dos partes:

- un *selector* - Un selector define dónde aplicar el estilo. 4D soporta los selectores "object type", "object name", "class", "all objects" y "attribute value".
- una *declaración* - La declaración define el estilo real a aplicar. Se pueden agrupar varias líneas de declaración para formar un bloque de declaración. Cada línea de un bloque de declaración CSS debe terminar con un punto y coma, y todo el bloque debe estar rodeado de llaves.

## Selectores de hojas de estilo

### Tipo de objeto

El tipo de objeto define el tipo de objeto al que hay que aplicar el estilo, y corresponde al selector de elementos CSS.

Especifique el tipo de objeto, luego entre llaves, declare el estilo o los estilos a aplicar.

El tipo objeto corresponde a la propiedad JSON [tipo](#) de los objetos formulario.

En el siguiente ejemplo, todos los objetos del tipo *botón* mostrarán el texto en la fuente Helvetica Neue, con un tamaño de 20 píxeles:

```
button {  
    font-family: Helvetica Neue;  
    font-size: 20px;  
}
```

Para aplicar el mismo estilo a varios tipos de objetos, especifique los tipos de objetos separados por un "," y luego, entre llaves, declare el o los estilos a aplicar:

```
text, input {  
    text-align: left;  
    stroke: grey;  
}
```

### Nombre del objeto

El nombre del objeto corresponde al selector de ID CSS y define un objeto específico al que hay que dar estilo, ya que el

nombre del objeto es único dentro del formulario.

Designe el objeto con un carácter "#" antes del nombre del objeto y, a continuación, entre llaves, declare el o los estilos a aplicar.

En el siguiente ejemplo, el texto del objeto con el nombre "okButton" se mostrará en fuente Helvetica Neue, con un tamaño de 20 píxeles:

```
#okButton {  
    font-family: Helvetica Neue;  
    font-size: 20px;  
}
```

## Class

Class corresponde al selector class CSS y define el estilo para todos los objetos formulario con el atributo `class`.

Puede especificar las clases a utilizar con un carácter "." seguido del nombre de la clase, y entre llaves, declarar el o los estilos a aplicar.

En el siguiente ejemplo, el texto de todos los objetos con el nombre de la clase `okButtons` se mostrará en la fuente Helvetica Neue, con un tamaño de 20 píxeles, alineado al centro:

```
.okButtons {  
    font-family: Helvetica Neue;  
    font-size: 20px;  
    text-align: center;  
}
```

Para indicar que un estilo debe aplicarse sólo a los objetos de un tipo determinado, especifique el tipo seguido de "." y el nombre de la clase, y luego, entre llaves, declare el estilo o los estilos a aplicar.

```
text.center {  
    text-align: center;  
    stroke: red;  
}
```

En la descripción del formulario 4D, se asocia un nombre de clase a un objeto mediante el atributo `class`. Este atributo contiene uno o varios nombres de clase, separados por un espacio:

```
class: "okButtons important"
```

## Todos los objetos

En correspondencia con el selector CSS universal, el carácter "\*" indica que el siguiente estilo se aplicará a todos los objetos del formulario.

Indique que un estilo debe aplicarse a todos los objetos formulario con el carácter "\*" y, a continuación, entre llaves, declare el o los estilos que deben aplicarse.

En el siguiente ejemplo, todos los objetos tendrán un fondo gris:

```
* {  
    fill: gray;  
}
```

## Atributos específicos

Los estilos correspondientes a los selectores de atributos CSS se pueden aplicar a todos los objetos formulario con un atributo específico.

Especifique el atributo entre corchetes y, a continuación, entre llaves, declare el estilo o los estilos a aplicar.

### Sintaxis soportadas

Sintaxis	Descripción
[attribute]	coincide con objetos con el <code>attribute</code>
[attribute="value"]	coincide con objetos cuyo valor del <code>attribute</code> contenga exactamente el "valor" especificado
[attribute~="value"]	coincide con los objetos con el valor del <code>attribute</code> que contiene el "valor" entre una lista de palabras separadas por espacios
[attribute = "value"]	coincide con objetos con un <code>attribute</code> cuyo valor empieza por "valor"

### Ejemplos

Todos los objetos con el atributo `borderStyle` tendrán líneas moradas:

```
[borderStyle]
{
    stroke: purple;
}
```

Todos los objetos de tipo texto con un atributo texto cuyo valor sea "Hello" tendrán letras azules:

```
text[text=Hello]
{
    stroke: blue;
}
```

Todos los objetos con un atributo texto cuyo valor sea "Hello" tendrán letras azules:

```
[text~="Hello"]
{
    stroke: blue;
}
```

Todos los objetos de tipo texto con un atributo texto cuyo valor comience por "Hello" tendrán letras azules:

```
text[text|=Hello]
{
    stroke: yellow;
}
```

## Declaraciones de hojas de estilo

### Media Queries

Media queries are used to apply color schemes to an application.

A media query is composed of a media feature and a value (e.g., `<media feature>:<value>` ).

Available media features:

- `prefers-color-scheme`

Available media feature expressions:

- light  
For using a light scheme
- dark  
For using a dark scheme

Los esquemas de color sólo son soportados en macOS.

## Ejemplo

Este CSS define una combinación de colores para el texto y el fondo del texto en el esquema claro (por defecto) y otra combinación cuando se selecciona el esquema oscuro:

```
@media (prefers-color-scheme: light) {  
  .textScheme {  
    fill: LightGrey;  
    stroke: Black;  
  }  
}  
  
@media (prefers-color-scheme: dark) {  
  .textScheme {  
    fill: DarkSlateGray;  
    stroke: LightGrey;  
  }  
}
```

## Atributos de objetos

La mayoría de los atributos del objeto formulario pueden ser definidos dentro de una hoja de estilo, excepto los siguientes atributos: - `method` - `type` - `class` - `evento` - `choiceList`, `excludedList`, `labels`, `list`, `requiredList` (tipo de lista)

Form object attributes can be declared with their [JSON name](#) as CSS attributes (not including object types, methods, events, and lists).

### Mapa de atributos

Los atributos listados a continuación pueden aceptar el nombre 4D o el nombre CSS.

4D	CSS
borderStyle	border-style
fill	background-color
fontFamily	font-family
fontSize	font-size
fontStyle	font-style
fontWeight	font-weight
stroke	color
textAlign	text-align
textDecoration	text-decoration
verticalAlign	vertical-align

Los valores específicos 4D (*por ejemplo*, `hundido`) no se soportan cuando se utilizan nombres de atributos CSS.

## Valores de atributos específicos

- Para los atributos `icon`, `picture` y `customBackgroundPicture` que soportan una ruta a una imagen, la sintaxis es:

```
icon: url("/RESOURCES/Images/Buttons/edit.png"); /* ruta absoluta */
icon: url("edit.png"); /* ruta relativa al archivo del formulario */
```

- Para `fill`, `stroke`, `alternateFill`, `horizontalLineStroke` y `verticalLineStroke`, se soportan tres sintaxis:
  - Nombre del color CSS: `fill: red;`
  - Valor hexadécimal: `fill: #FF0000;`
  - función `rgb()`: `fill:rgb(255,0,0)`
- Si una cadena utiliza caracteres prohibidos en CSS, puede rodear la cadena con comillas simples o dobles. Por ejemplo:
  - una referencia xliff: `tooltip: ":xiff:CommonMenuFile";`
  - un datasource con la expresión de campo: `dataSource: "[Table_1:1]ID:1";`

## Orden de prioridad

Los proyectos 4D priorizan las definiciones de estilo en conflicto, primero por la definición del formulario y luego por las hojas de estilo.

## JSON vs Hoja de estilo

Si un atributo está definido en la descripción del formulario JSON y en una hoja de estilo, 4D utilizará el valor del archivo JSON.

Para anular este comportamiento, el valor del estilo debe ir seguido de una declaración `!important`.

Ejemplo 1:

Descripción del formulario JSON	Hoja de estilo	4D muestra
<code>"text": "Button",</code>	<code>text: Edit;</code>	<code>"Button"</code>

## Ejemplo 2:

Descripción del formulario JSON	Hoja de estilo	4D muestra
"text": "Button",	text: Edit !important;	"Edit"

## Hojas de estilo múltiples

Durante la ejecución, 4D prioriza automáticamente las hojas de estilo en el siguiente orden:

1. El formulario 4D cargará primero el archivo CSS por defecto `/SOURCES/styleSheets.css`.
2. Luego cargará el archivo CSS para la plataforma actual `/SOURCES/styleSheets_mac.css` o `/SOURCES/styleSheets_windows.css`.
3. Si existe, entonces cargará un archivo CSS específico definido en el formulario JSON:

- un archivo para ambas plataformas:

```
"css": "<path>"
```

- o una lista de archivos para ambas plataformas:

```
"css": [  
    "<path1>",  
    "<path2>"  
,
```

- o una lista de archivos por plataforma:

```
"css": [  
    {"path": "<path>", "media": "mac"},  
    {"path": "<path>", "media": "windows"},  
,
```

Las rutas de los archivos pueden ser relativas o absolutas. \* Las rutas relativas se resuelven en relación con el archivo de descripción del formulario JSON. \* Por razones de seguridad, sólo se aceptan las rutas del sistema de archivos para las rutas absolutas. (e.g., "/RESOURCES", "/DATA")

## Ver también

Ver la presentación en video [CSS for 4D Forms](#).

# Imágenes

4D soporta específicamente las imágenes utilizadas en sus formularios.

## Formatos nativos soportados

4D integra la gestión nativa de los formatos de imagen. Esto significa que las imágenes se mostrarán y almacenarán en su formato original, sin ninguna interpretación en 4D. Las especificidades de los diferentes formatos (sombreado, áreas transparentes, etc.) se mantendrán cuando se copien y peguen, y se mostrarán sin alteraciones. Este soporte nativo es válido para todas las imágenes almacenadas en los formularios de 4D: [imágenes estáticas](#) pegadas en el modo Diseño, imágenes pegadas en [objetos de entrada](#) en ejecución, etc.

Los formatos de imagen más comunes son soportados en ambas plataformas: .jpeg, .gif, .png, .tiff, .bmp, etc. En macOS, el formato .pdf también está disponible para su codificación y descodificación.

La lista completa de formatos soportados varía según el sistema operativo y los códecs personalizados que estén instalados en las máquinas. Para saber qué códecs están disponibles, debe utilizar el comando `PICTURE CODEC LIST` (ver también la descripción de [tipo de datos imagen](#)).

## Formato de imagen no disponible

Se muestra un ícono específico para las imágenes guardadas en un formato que no está disponible en la máquina. La extensión del formato que falta se muestra en la parte inferior del ícono:



El ícono se utiliza automáticamente en todos los lugares en los que se pretende visualizar la imagen:

FirstName :	LastName :	Photo :
Elizabeth	Smith	
Gerry	Mc Namara	
Henry	Portier	

Este ícono indica que la imagen no puede ser visualizada o manipulada localmente, pero puede ser guardada sin alteraciones para que pueda ser visualizada en otras máquinas. Este es el caso, por ejemplo, para las imágenes PDF en Windows, o las imágenes en formato PICT.

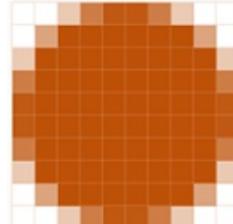
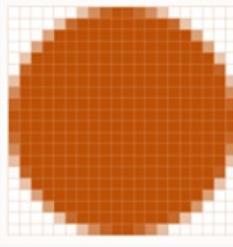
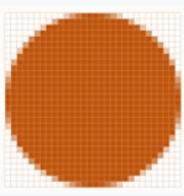
## Imágenes de alta resolución

4D soporta imágenes de alta resolución tanto en plataformas macOS como Windows. Las imágenes de alta resolución pueden definirse por el factor de escala o dpi.

### Factor de escala

Las pantallas de alta resolución tienen una mayor densidad de píxeles que las pantallas estándar tradicionales. Para que las imágenes se muestren correctamente en pantallas de alta resolución, el número de píxeles de la imagen debe multiplicarse por el *factor de escala* (es decir, dos veces más grande, tres veces más grande, etc.).

Cuando se utilizan imágenes de alta resolución, se puede especificar el factor de escala añadiendo "@nx" en el nombre de la imagen (donde  $n$  designa el factor de escala). En la tabla siguiente, puede ver que el factor de escala se indica en los nombres de las imágenes de alta resolución, *circle@2x.png* y *circle@3x.png*.

Tipo de visualización	Factor de escala	Ejemplo	
Resolución estándar	densidad de pixel 1:1.	1x 	<i>circle.png</i>
Alta resolución	La densidad de píxeles se ha multiplicado por 2 o 3.	2x 	3x  <i>circle@2x.png</i> <i>circle@3x.png</i>

Las imágenes de alta resolución con la convención @nx pueden utilizarse en los siguientes objetos:

- [Imágenes estáticas](#)
- [Botones/radio/casillas de selección](#)
- [Botones imagen/imagen Pop-up](#)
- [Pestañas](#)
- [Encabezados de list box](#)
- [Iconos del menú](#)

4D prioriza automáticamente las imágenes con mayor resolución. 4D prioriza automáticamente las imágenes con mayor resolución. Incluso si un comando o propiedad especifica *circle.png*, se utilizará *circle@3x.png* (si existe).

Tenga en cuenta que la priorización de la resolución sólo se produce para la visualización de imágenes en pantalla, no se realiza una priorización automática al imprimir.

## DPI

Aunque 4D prioriza automáticamente la resolución más alta, existen, sin embargo, algunas diferencias de comportamiento en función de los ppp de la pantalla y de la imagen(\*), y del formato de la imagen:

Operación	Comportamiento
Soltar o pegar	<p>Si la imagen tiene:</p> <ul style="list-style-type: none"> <li>• 72dpi o 96dpi - La imagen tiene el formato "Center" y el objeto que contiene la imagen tiene el mismo número de píxeles.</li> <li>• Otro dpi - La imagen tiene el formato "{Escalada para encajar} (/docs/Rx/es/FormObjects/propertiesPicture.html#scaled-to-fit)" y el objeto que contiene la imagen es igual a <math>(\text{número de píxeles de la imagen} * \text{dpi de la pantalla}) / (\text{dpi de la imagen})</math></li> <li>• Sin dpi - La imagen tiene el formato "{Escala para ajustar} (/docs/Rx/es/FormObjects/propertiesPicture.html#scaled-to-fit)".</li> </ul>
Tamaño automático (menú contextual del editor de formularios)	<p>Si el formato de visualización de la imagen es:</p> <ul style="list-style-type: none"> <li>• Scaled - El objeto que contiene la imagen se redimensiona según <math>(\text{número de píxeles de la imagen} * \text{dpi de la pantalla}) / (\text{dpide la imagen})</math></li> <li>• Sin escalar - El objeto que contiene la imagen tiene el mismo número de píxeles que la imagen.</li> </ul>

(\*) Generalmente, macOS = 72 dpi, Windows = 96 dpi

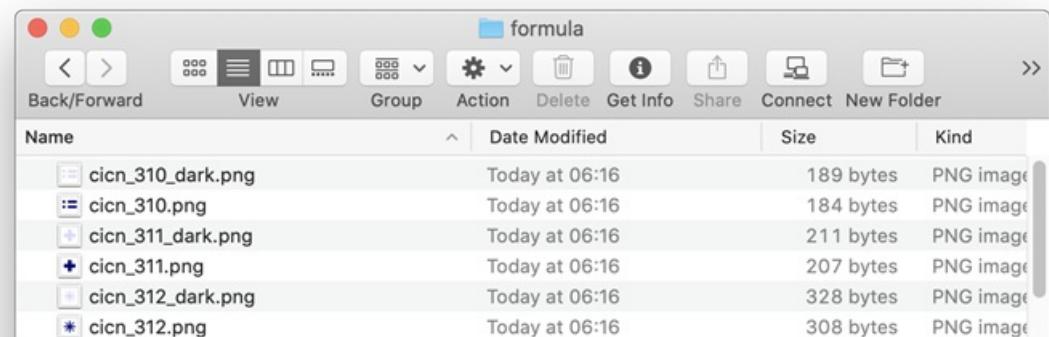
## Imágenes en modo oscuro (sólo en macOS)

Puede definir imágenes e iconos específicos que se utilizarán en lugar de las imágenes estándar cuando [los formularios utilicen el esquema oscuro](#).

Una imagen en modo oscuro se define de la siguiente manera:

- la imagen en modo oscuro tiene el mismo nombre que la versión estándar (modo claro) con el sufijo " \_dark "
- la imagen en modo oscuro se almacena junto a la versión estándar.

En tiempo de ejecución, 4D cargará automáticamente la imagen clara u oscura según el [modo de colores de formulario actual](#).



## Coordenadas del ratón en una imagen

4D permite recuperar las coordenadas locales del ratón en un [objeto de entrada](#) asociado a una [expresión de imagen](#), en caso de que se haga clic o se pase por encima, incluso si se ha aplicado un desplazamiento o zoom a la imagen. Este mecanismo, similar al de un mapa de imágenes, puede utilizarse, por ejemplo, para manejar barras de botones desplazables o la interfaz de un software de cartografía.

Las coordenadas se devuelven en las [MouseX](#) and [MouseY Variables Sistema](#). Las coordenadas se expresan en píxeles con respecto a la esquina superior izquierda de la imagen (0,0). Si el ratón está fuera del sistema de coordenadas de la imagen, se devuelve -1 en [MouseX](#) y [MouseY](#).

Puede obtener el valor de estas variables como parte de los eventos formulario [On Clicked](#), [On Double Clicked](#), [On Mouse up](#), [On Mouse Enter](#), o [On Mouse Move](#).



# Lista de propiedades JSON

Esta página ofrece una lista completa de todas las propiedades de los formularios, ordenadas por su nombre JSON. Haga clic en el nombre de una propiedad para acceder a su descripción detallada.

En el capítulo "Propiedades del formulario", las propiedades se ordenan según sus nombres y temas en la lista de propiedades.

a - c - d - e - f - h - i - m - p - r - s - w

Propiedad	Descripción	Valores posibles
a		
<code>bottomMargin</code>	Valor del margen vertical (en píxeles)	mínimo: 0
c		
<code>colorScheme</code>	Esquema de colores para el formulario	"dark", "light"
d		
<code>destination</code>	Tipo de formulario	"detailScreen", "listScreen", "detailPrinter", "listPrinter"
e		
<code>entryOrder</code>	El orden en el cual los objetos activos son seleccionados cuando la tecla Tab o la tecla Retorno de carro se utilizan en un formulario de entrada	Colección de nombres de objetos 4D Form
<code>events</code>	Lista de todos los eventos seleccionados para el objeto o el formulario	Colección de nombres de eventos, por ejemplo ["onClick", "onDataChange"...].
f		
<code>formSizeAnchor</code>	Nombre del objeto cuya posición determina el tamaño del formulario. (longitud mínima: 1)	Nombre de un objeto 4D
h		
<code>height</code>	Altura del formulario	mínimo: 0
i		
<code>inheritedForm</code>	Designa el formulario a heredar	Nombre (cadena) de la tabla o formulario proyecto O una ruta POSIX (cadena) a un archivo .json que describa el formulario O un objeto que describa el formulario
<code>inheritedFormTable</code>	Designa la tabla que utilizará un formulario heredado	Un nombre o número de tabla
m		
<code>markerBody</code>	Posición del marcador de detalle	mínimo: 0
<code>markerBreak</code>	Posición del marcador de ruptura	mínimo: 0
<code>markerFooter</code>	Posición del marcador de pie	mínimo: 0
<code>markerHeader</code>	Posición del marcador de encabezado	integer minimum: 0; integer array minimum: 0
<code>memorizeGeometry</code>	Guarda los parámetros del formulario	true, false

<code>memorizeGeometry</code>	Guarda los parámetros del formulario Desarrolla y arrastra la ventana del mismo	true, false Valores posibles
<code>menuBar</code>	Barra de menú a asociar al formulario	Nombre de una barra de menú válida
<code>method</code>	Un nombre de método proyecto.	El nombre de un método proyecto existente
<code>p</code>		
<code>pages</code>	Colección de páginas (cada página es un objeto)	Objetos página
<code>pageFormat</code>	objeto	Propiedades de impresión disponibles
<code>r</code>		
<code>rightMargin</code>	Valor del margen horizontal (en píxeles)	mínimo: 0
<code>s</code>		
<code>shared</code>	Especifica si un formulario puede utilizarse como subformulario	true, false
<code>w</code>		
<code>ancho</code>	Ancho del formulario	mínimo: 0
<code>windowMaxHeight</code>	La mayor altura permitida de la ventana del formulario	mínimo: 0
<code>windowMaxWidth</code>	El mayor ancho permitido de la ventana de formulario	mínimo: 0
<code>windowMinHeight</code>	La menor altura permitida de la ventana de formulario	mínimo: 0
<code>windowMinWidth</code>	El menor ancho permitido de la ventana de formulario	mínimo: 0
<code>windowSizingX</code>	Dimensionamiento vertical de la ventana del formulario	"fixed", "variable"
<code>windowSizingY</code>	Dimensionamiento horizontal de la ventana del formulario	"fixed", "variable"
<code>windowTitle</code>	Designa el título de una ventana de formulario	Un nombre para la ventana de formulario

# Acción

## Método

Referencia de un método adjunto al formulario. Puede utilizar un método formulario para gestionar datos y objetos, pero generalmente es más sencillo y eficiente utilizar un método objeto para estos fines. Ver [Métodos especializados](#).

No llame a un método formulario - 4D lo llama automáticamente cuando un evento implica el formulario al que el método está asociado.

Se soportan varios tipos de referencias de métodos:

- una ruta de archivo de método proyecto estándar, es decir, que utilice el siguiente modelo:

`method.4dm`

Este tipo de referencia indica que el archivo de método se encuentra en la ubicación por defecto ("sources/{TableForms/numTable} | {Forms}/formName/"). En este caso, 4D maneja automáticamente el método formulario cuando se ejecutan operaciones en el formulario (renombrar, duplicar, copiar/pegar...)

- un nombre de método proyecto: nombre de un método proyecto existente sin extensión de archivo, es decir:

`myMethod` En este caso, 4D no soporta automáticamente las operaciones de formulario.

- una ruta de acceso al archivo del método personalizado que incluya la extensión .4dm, por ejemplo:

`MyMethods/myFormMethod.4dm` También puede utilizar un sistema de archivos:

`/RESOURCES/Forms/FormMethod.4dm` En este caso, 4D no ofrece soporte automático para las operaciones con objetos.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
method	texto	Ruta estándar o personalizada del método formulario, o nombre del método proyecto

# Propiedades de los formularios

---

## Esquema de colores

La propiedad de esquema de color sólo se aplica en macOS.

This property defines the color scheme for the form. This property defines the color scheme for the form. This can be changed for the form to one of the following two options:

- dark -- texto claro sobre fondo oscuro
- light - texto oscuro en un fondo claro > A defined color scheme can not be overridden by a CSS.

El número de caracteres para el título de una ventana está limitado a 31.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
colorScheme	cadena	"dark", "light"

## Pages

Cada formulario consta de al menos dos páginas:

- una página 0 (página de fondo)
- una página 1 (página principal)

Para más información, consulte [Páginas formulario](#).

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
pages	colección	Colección de páginas (cada página es un objeto, la página 0 es el primer elemento)

## Nombre del formulario

This property is the name of the form itself and is used to refer to the form by name using the 4D language. The form name must comply with the [rules specified for identifiers](#) in 4D.

### Gramática JSON

The form name is defined by the name of the folder that contains the form.4Dform file. See [project architecture](#) for more information.

## Tipo de formulario

The form type, *i.e.* its destination, defines the features that will be available to the form. For example, [markers](#) can only

be set for list (output) table forms.

Each table in a database generally has at least two table forms. One for listing records on-screen and the other for displaying one record at a time (used for data entry and modifications):

- Output form - the *output form* or *list form* displays a list of records, with a single line per record. The results of queries are shown in an output form and users can double-click a line to display the input form for that record.

ID :	name :
1	Friends
3	Work
4	Personal
5	Family

- Formulario de entrada - utilizado para la entrada de datos. It displays a single record per screen and typically has buttons for saving and canceling modifications to the record and for navigating from record to record (*i.e.*, First Record, Last Record, Previous Record, Next Record).

Los tipos soportados dependen de la categoría de formulario:

Tipo de formulario	Gramática JSON	Descripción	Soportado con
Formulario detallado	detailScreen	Un formulario de visualización para introducir y modificar datos	Formularios proyecto - Formularios tabla
Formulario detallado imprimible	detailPrinter	A printed report with one page per record, such as an invoice	Formularios proyecto - Formularios tabla
Formulario listado	listScreen	Un formulario para listar los registros en la pantalla	Formularios tabla
Formulario de lista imprimible	listPrinter	A printed report that lists records	Formularios tabla
Ninguno	<i>no destination</i>	A form with no specific feature	Formularios proyecto - Formularios tabla

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
destination	cadena	"detailScreen", "listScreen", "detailPrinter", "listPrinter"

## Nombre del formulario heredado

This property designates the [form to inherit](#) in the current form.

To inherit from a table form, set the table in the [Inherited Form Table](#) property.

To remove inheritance, select <None> in the Property List (or " " in JSON).

Gramática JSON

Nombre	Tipos de datos	Valores posibles
inheritedForm	cadena	Name of table or project form OR a POSIX path to a .json file describing the form OR an object describing the form

## Tablas de formulario heredadas

This property specifies the database table from which to [inherit a form](#) in the current form.

Set to <None> in the Property List (or " " in JSON) to inherit from a project form.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
inheritedFormTable	string o number	nombre de tabla o número de tabla

## Publicado como Subformulario

Para que un formulario componente sea seleccionado como [subformulario](#) en una aplicación anfitriona, debe haber sido compartido explícitamente. When this property is selected, the form will be published in the host application.

Only project forms can be specified as published subforms.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
shared	booleano	true, false

## Memorizar geometría

Cuando se utiliza esta opción, si la ventana se abre utilizando el comando `Open form window` con el parámetro `*`, varios parámetros del formulario son guardados automáticamente por 4D cuando se cierra la ventana, independientemente de cómo se hayan modificado durante la sesión:

- la página actual,
- la posición, el tamaño y la visibilidad de cada objeto del formulario (incluyendo el tamaño y la visibilidad de las columnas de list box).

Esta opción no tiene en cuenta los objetos generados con el comando `OBJECT DUPLICATE`. Para que un usuario pueda recuperar su entorno al utilizar este comando, el desarrollador debe repetir la secuencia de creación, definición y posicionamiento de los objetos.

Cuando se selecciona esta opción, la opción [Guardar valor](#) está disponible para ciertos objetos.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
memorizeGeometry	booleano	true, false

Ver también

[Guardar valor](#)

## Título de la ventana

El título de la ventana se utiliza cuando se abre el formulario mediante los comandos `Open form window` y `Open window 4D` en el entorno de la aplicación. El nombre de la ventana aparece en la barra de título de la ventana.

Puede utilizar referencias dinámicas para definir los nombres de ventana de los formularios, es decir:

- Una referencia estándar XLIFF almacenada en la carpeta Resources.
- Una etiqueta de tabla o de campo: la sintaxis a aplicar es `<?[TableName]FieldNum>` o
- Una variable o un campo: la sintaxis a aplicar es `<[VariableName>` o `<[TableName]FieldName>`. El valor actual del campo o de la variable se mostrará en el título de la ventana.

El número de caracteres para el título de una ventana está limitado a 31.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
windowTitle	cadena	El nombre de la ventana como texto plano o de referencia

# Tamaño formulario

4D permite definir el tamaño tanto del formulario como de la [ventana](#). Estas propiedades son interdependientes y su interfaz de aplicación es el resultado de su interacción.

Las opciones de tamaño dependen del valor de la opción Tamaño basado en.

## Tamaño basado en

- Tamaño automático: el tamaño del formulario será el necesario para mostrar todos los objetos, al que se añadirán los valores de margen (en píxeles) introducidos en

\*\*los campos *Margen hor.* y *Margen ver.* *Margin fields.*

Puede elegir esta opción cuando desee utilizar objetos activos situados en un área fuera de la pantalla (es decir, fuera del rectángulo delimitador de la ventana) con una ventana de tamaño automático. Gracias a esta opción, la presencia de estos objetos no modificará el tamaño de la ventana.

- Definir tamaño: el tamaño del formulario se basará en lo que introduzca (en píxeles) en los campos [Ancho](#) y [Alto](#).
- <object name>: el tamaño del formulario se basará en la posición del objeto formulario seleccionado. Por ejemplo, si elige un objeto situado en la parte inferior derecha del área a mostrar, el tamaño del formulario consistirá en un rectángulo cuya esquina superior izquierda será el origen del formulario y la esquina inferior derecha corresponderá a la del objeto seleccionado, más los valores de los márgenes.

Para los formularios de salida, sólo se pueden utilizar los campos Margen \*\*hor.\*\* o [Largo](#) son disponibles.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
formSizeAnchor	cadena	Nombre del objeto a utilizar para definir el tamaño del formulario

## Altura

\*\*los campos \*\*Margen *hor.* y *Margen ver.* *Margin fields.*

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
height	number	valor entero largo

## Margen hor.

Tamaño automático: el tamaño del formulario será el necesario para mostrar todos los objetos, al que se añadirán los valores de margen (en píxeles) introducidos en

Altura del formulario (en píxeles) cuando el [tamaño del formulario](#) está definido en [\\*\\*Fijar tamaño\\*\\*](#).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
rightMargin	number	valor entero largo

---

## Margen hor.

Valor a añadir (en píxeles) al margen derecho del formulario cuando el [tamaño del formulario](#) está definido en Tamaño automático o <object name>

Este valor también determina los márgenes derechos de los formularios utilizados en el editor de etiquetas.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
bottomMargin	number	valor entero largo

---

## Ancho

Valor a añadir (en píxeles) al margen inferior del formulario cuando el [tamaño del formulario](#) está definido en Tamaño automático o <object name>.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
ancho	number	valor entero largo

# Marcadores

Estas propiedades permiten especificar la ubicación precisa de los marcadores en la regla vertical de un formulario tabla. Los marcadores se utilizan principalmente en los formularios de salida. They control the information that is listed and set header, breaks, detail and footer areas of the form. Any object that placed in these areas is displayed or printed at the appropriate location.

Whenever any form is used for output, either for screen display or printing, the output marker lines take effect and the areas display or print at designated locations. The markers also take effect when a form is used as the List form in a subform area. However, they have no effect when a form is used for input.

Methods that are associated with objects in these areas are executed when the areas are printed or displayed as long as the appropriate events have been activated. For example, a object method placed in the Header area is executed when the `On Header` event takes place.

---

## Form Break

Form Break areas are displayed once at the end of the list of records and are printed once after the records have been printed in a report.

The Break area is defined as the area between the Detail control line and the Break control line. There can be [several Break areas](#) in your report.

You can make Break areas smaller or larger. You can use a Break area to display information that is not part of the records (instructions, current date, current time, etc.), or to display a line or other graphic element that concludes the screen display. In a printed report, you can use a Break area for calculating and printing subtotals and other summary calculations.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
markerBreak	integer   integer collection	Break marker position or collection of break marker positions in pixels. Minimum value: 0

---

## Formulario detallado

The form Detail area is displayed on the screen and printed once for each record in a report. The Detail area is defined as the area between the Header control line and the Detail control line.

Puede hacer el área Detalle más pequeña o más grande. Whatever you place in the Detail area is displayed or printed once for each record. Most often you place fields or variables in the Detail area so that the information in each record is displayed or printed, but you can place other elements in the Detail area as well.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
markerBody	integer	Posición del marcador de detalle. Mínimo: 0

---

## Pie del formulario

The Form Footer area is displayed on screen under the list of records. It is always printed at the bottom of every page of a report. The Footer area is defined as the area between the Break control line and the Footer control line. Puede

hacer que el área del pie de página sea más pequeña o más grande.

You can use the Footer area to print graphics, page numbers, the current date, or any text you want at the bottom of each page of a report. For output forms designed for use on screen, the Footer area typically contains buttons that give the user options such as doing a search or sort, printing records, or putting away the current report. Se aceptan los objetos activos.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
markerFooter	integer	mínimo: 0

## Encabezado del formulario

The form Header area is displayed at the top of each screen and is printed at the top of each page of a report. The Header area is defined as the area above the Header control line. Puede hacer el área Encabezado más pequeña o más grande. You can use the Header area for column names, for instructions, additional information, or even a graphic such as a company logo or a decorative pattern.

You can also place and use active objects in the Header area of output forms displayed as subforms, in the records display window or using the `DISPLAY SELECTION` and `MODIFY SELECTION` commands. Se pueden insertar los siguientes objetos activos:

- Botones, botones imagen,
- Combo boxes, drop-down lists, picture pop-up menus,
- listas jerárquicas, list boxes
- Botones de radio, casillas de selección, casillas de selección 3D,
- Progress indicators, rulers, steppers, spinners.

Standard actions such as `Add Subrecord`, `Cancel` (lists displayed using `DISPLAY SELECTION` and `MODIFY SELECTION`) or `Automatic splitter` can be assigned to the inserted buttons. The following events apply to the active objects you insert in the Header area: `On Load`, `On Clicked`, `On Header`, `On Printing Footer`, `On Double Clicked`, `On Drop`, `On Drag Over`, `On Unload`. Keep in mind that the form method is called with the `On Header` event after calling the object methods of the area.

The form can contains [additional header areas](#) to be associated with additional breaks. A level 1 Header is printed just before the records grouped by the first sorted field are printed.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
markerHeader	integer   integer collection	Header marker position or collection of header marker positions in pixels. Minimum value: 0

## Áreas adicionales

You can create additional Break areas and Header areas for a report. These additional areas allow you to print subtotals and other calculations in a report and to display other information effectively.

Additional areas are defined when you use a collection of positions in the [Form Break](#) and [Form Header](#) properties.

In the 4D Form editor, you create additional control lines by holding down the Alt key while clicking the appropriate control marker.

A form always starts with a Header, Detail, Break level 0, and Footer areas.

Break at level 0 zero takes in all the records; it occurs after all the records are printed. Additional Break areas can be added, i.e. a Break level 1, Break level 2, etc.

A Break level 1 occurs after the records grouped by the first sorted field are printed.

Etiqueta	Descripción	Prints after groups created by:
B1	Nivel de ruptura 1	Primer campo ordenado
B2	Nivel de ruptura 2	Segundo campo ordenado
B3	Nivel de ruptura 3	Tercer campo ordenado

Additional Header areas are associated with Breaks. A level 1 Header is printed just before the records grouped by the first sorted field are printed.

Etiqueta	Descripción	Prints after groups created by:
H1	Encabezado en el nivel 1	Primer campo ordenado
H2	Encabezado en el nivel 2	Segundo campo ordenado
H3	Encabezado en el nivel 3	Tercer campo ordenado

If you use the `Subtotal` function to initiate Break processing, you should create a Break area for every level of Break that will be generated by the sort order, minus one. If you do not need anything printed in one of the Break areas, you can reduce its size to nothing by placing its marker on top of another control line. If you have more sort levels than Break areas, the last Break area will be repeated during printing.

# Menú

## Barra de menús asociada

Cuando se asocia una barra de menú a un formulario, ésta se añade a la derecha de la barra de menú actual cuando el formulario se muestra en el entorno Aplicación.

La selección de un comando de menú hace que se envíe un evento `On Menu Selected` al método formulario; entonces puede utilizar el comando `Menu selected` para probar el menú seleccionado.

Si la barra de menús del formulario es idéntica a la barra de menús actual, no se añade.

La barra de menús del formulario funcionará tanto para los formularios de entrada como para los de salida.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
menuBar	cadena	Nombre de una barra de menú

# Imprimir

## Settings

Permite definir los parámetros de impresión específicos para el formulario. Esta funcionalidad es útil para ver los límites de páginas de impresión en el editor de formularios.

Compatibilidad: aunque estos parámetros se tengan en cuenta cuando se imprime el formulario en modo Aplicación, se desaconseja confiar en esta funcionalidad para almacenar los parámetros de impresión del formulario, debido a las limitaciones relativas a la plataforma y al driver. Es muy recomendable utilizar los comandos 4D `Print settings to BLOB / BLOB to print settings` que son más poderosos.

Puede modificar los siguientes parámetros de impresión:

- Formato del papel
- Orientación del papel
- Escala de la página

Las opciones disponibles dependen de la configuración del sistema.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
pageFormat	objeto	Propiedades de impresión disponibles: <code>paperName</code> , <code>paperWidth</code> , <code>paperHeight</code> , <code>orientation</code> , <code>scale</code>
paperName	cadena	"A4", "US Letter"...
paperWidth	cadena	Utilizado si no se encuentra un papel llamado <code>paperName</code> . Requiere sufijo de unidad: <code>pt</code> , <code>in</code> , <code>mm</code> , <code>cm</code> .
paperHeight	cadena	Utilizado si no se encuentra un papel llamado <code>paperName</code> . Requiere sufijo de unidad: <code>pt</code> , <code>in</code> , <code>mm</code> , <code>cm</code> .
orientation	cadena	"landscape" (por defecto es "portrait")
scale	number	mínimo: 0

# Tamaño de la ventana

## Alto fijo

Si selecciona esta opción, la altura de la ventana quedará bloqueada y el usuario no podrá cambiar su tamaño.

Si no se selecciona esta opción, se puede modificar el ancho de la ventana del formulario. En este caso, las propiedades [Altura mínima](#) y [Altura máxima](#) pueden utilizarse para determinar los límites de redimensionamiento.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
windowSizingY	cadena	"fixed", "variable"

---

## Ancho fijo

Si selecciona esta opción, el ancho de la ventana quedará bloqueada y el usuario no podrá cambiar su tamaño.

Si no se selecciona esta opción, se puede modificar el ancho de la ventana del formulario. En este caso, las propiedades [Ancho mínimo](#) y [Ancho máximo](#) pueden utilizarse para determinar los límites de redimensionamiento.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
windowSizingX	cadena	"fixed", "variable"

---

## Altura máxima, Altura mínima

Altura máxima y mínima (en píxeles) de una ventana de formulario redimensionable si la opción [Alto fijo](#) no está definida.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
windowMinHeight	number	valor entero largo
windowMaxHeight	number	valor entero largo

---

## Ancho máximo, Ancho mínimo

Ancho máximo y mínimo (en píxeles) de una ventana de formulario redimensionable si la opción [Ancho fijo](#) no está definida.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
windowMinWidth	number	valor entero largo
windowMaxWidth	number	valor entero largo



# Acerca de los objetos formularios 4D

Usted crea y personaliza los formularios de su aplicación manipulando los objetos que contienen. Puede añadir objetos, repositionar objetos, definir propiedades de los objetos, aplicar reglas de negocio especificando restricciones de entrada de datos o escribir métodos de objetos que se ejecuten automáticamente cuando se utilice el objeto.

## Objetos activos y estáticos

Los formularios 4D soportan una gran cantidad de objetos activos y estáticos integrados:

- Los objetos activos realizan una tarea o una función de la interfaz. Los campos son objetos activos. Los otros objetos activos -objetos editable (variables), combo box, listas desplegables, botones imagen, etc.- almacenan los datos temporalmente en la memoria o realizan alguna acción, como abrir una caja de diálogo, imprimir un informe o iniciar un proceso en segundo plano.
- Los objetos estáticos se utilizan generalmente para definir la apariencia del formulario y sus etiquetas, así como para la interfaz gráfica. Los objetos estáticos no tienen variables asociadas como los objetos activos. Sin embargo, se pueden insertar objetos dinámicos en objetos estáticos.

## Gestión de objetos de formulario

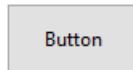
Puede añadir o modificar objetos formulario 4D de las siguientes maneras:

- [Editor de formularios](#): arrastre un objeto de la barra de herramientas del editor de formularios al formulario. A continuación, utilice la lista de propiedades para especificar las propiedades del objeto. Consulte el capítulo [Construcción de formularios](#) para más información.
- 4D language: los comandos del tema [Objetos \(Formularios\)](#) como `OBJECT DUPLICATE` o `OBJECT SET FONT STYLE` permiten crear y definir objetos de formulario.
- Código JSON en formularios dinámicos: define las propiedades utilizando JSON. Utilice la propiedad [tipo](#) para definir el tipo de objeto y, a continuación, indique sus propiedades disponibles. Ver la página [Formularios dinámicos](#) para obtener información.

Ejemplo de un objeto botón:

# Botón

Un botón es un objeto activo al que se le puede asignar una acción (*por ejemplo*, una tarea de base de datos o una función de interfaz) para que la realice cuando un usuario haga clic en él.



Los botones pueden cumplir diversas funciones, según su estilo y la acción que se les asigne. Por ejemplo, los botones pueden guiar al usuario a través de un cuestionario o formulario para que lo llene, o para que tome decisiones. Dependiendo de sus propiedades, un botón puede estar diseñado para ser presionado una sola vez y ejecutar un comando, mientras que otros pueden requerir que el usuario haga clic más de una vez para recibir el resultado deseado.

## Gestión de botones

Las acciones asignadas a los botones pueden provenir de [acciones estándar](#) o de métodos de objetos personalizados. Algunos ejemplos de acciones típicas son permitir al usuario aceptar, cancelar o eliminar registros, copiar o pegar datos, pasar de una página a otra en un formulario de varias páginas, abrir, eliminar o añadir registros en un subformulario, manejar los atributos de las fuentes en las áreas de texto, etc.

Los botones con acciones estándar se atenúan cuando es apropiado durante la ejecución del formulario. Por ejemplo, si se muestra el primer registro de una tabla, un botón con la acción estándar `firstRecord` aparecería atenuado.

Si desea que un botón realice una acción que no está disponible como acción estándar, deje el campo de acción estándar vacío y escriba un método de objeto para especificar la acción del botón. Para más información sobre los métodos de objetos y cómo crearlos y asociarlos, ver [Uso de los métodos objeto](#). Normalmente, se activaría el evento `On Clicked` y el método se ejecutaría sólo cuando se presiona el botón. Puede asociar un método a cualquier botón.

La [variable](#) asociada a un botón se define automáticamente a 0 cuando el formulario se ejecuta por primera vez en modo Diseño o Aplicación. Cuando el usuario hace clic en un botón, su variable se define como 1.

A un botón se le puede asignar tanto una acción estándar como un método. En este caso, si el botón no está desactivado por la acción estándar, el método se ejecuta antes de la acción estándar.

## Estilos de botón

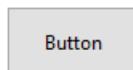
Los estilos de botón controlan la apariencia general de un botón, así como sus propiedades disponibles. Es posible aplicar diferentes estilos predefinidos a los botones o asociarles menús emergentes. Se puede obtener un gran número de variaciones combinando estas propiedades/comportamientos.

Con la excepción de las [propiedades-disponibles](#), muchos objetos botón son *estructuralmente* idénticos. La diferencia está en el tratamiento de sus variables asociadas.

4D ofrece botones en los siguientes estilos predefinidos:

### Clásico

El estilo de botón Clásico es un botón sistema estándar (*es decir*, un rectángulo con una etiqueta descriptiva) que ejecuta el código cuando el usuario hace clic en él.



Por defecto, el estilo Clásico tiene un fondo gris claro con una etiqueta en el centro. Cuando el cursor pasa por encima del estilo de botón Clásico, el borde y el color de fondo cambian para demostrar que tiene el foco. Además de iniciar la ejecución del código, el estilo del botón Clásico imita un botón mecánico cambiando rápidamente el color de fondo al ser presionado.

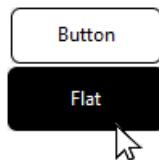
Ejemplo JSON:

```
"miBotón": {  
    "tipo": "button", //define el tipo de objeto  
    "style": "regular", //define el estilo del botón  
    "defaultButton": "true" //define el botón como opción por defecto  
    "text": "OK", //texto que aparecerá en el botón  
    "action": "Cancel", //acción a realizar  
    "left": 60, //posición izquierda en el formulario  
    "top": 160, //posición superior en el formulario  
    "width": 100, //ancho del botón  
    "height": 20 //altura del botón  
}
```

Sólo los estilos Clásico y Plano ofrecen la propiedad [Botón por defecto](#).

## Plano

El estilo de botón Plano es un botón sistema estándar (\*es decir, \*, un rectángulo con una etiqueta descriptiva) que ejecuta código cuando un usuario hace clic en él.



Por defecto, el estilo Plano tiene un fondo blanco con una etiqueta en el centro, esquinas redondeadas y una apariencia minimalista. El estilo gráfico del botón Flat es especialmente útil para los formularios a imprimir.

Ejemplo JSON:

```
"myButton": {  
    "type": "button",  
    "style": "flat",  
    "defaultButton": "true"  
    "text": "OK",  
    "action": "Cancel",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

Sólo los estilos Clásico y Plano ofrecen la propiedad [Botón por defecto](#).

## Barra de herramientas

El estilo de botón de la barra de herramientas está destinado principalmente a integrarse en una barra de herramientas. Incluye la opción de añadir un menú emergente (indicado por un triángulo invertido) que generalmente se utiliza para mostrar opciones adicionales para que el usuario las seleccione.

Por defecto, el estilo Barra de herramientas tiene un fondo transparente con una etiqueta en el centro. La apariencia del botón puede ser diferente cuando el cursor pasa por encima de él dependiendo del sistema operativo:

- *Windows* - el botón se resalta cuando utiliza la propiedad "Con menú emergente", se muestra un triángulo a la derecha y en el centro del botón.



- *macOS* - el resalte del botón nunca aparece. Cuando utiliza la propiedad "Con menú emergente", aparece un triángulo a la derecha y en la parte inferior del botón.

Ejemplo JSON:

```
"myButton": {  
    "type": "button",  
    "style": "toolbar",  
    "text": "OK",  
    "popupPlacement": "separated",  
    "action": "Cancel",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

## Bevel

El estilo de botón Bisel combina la apariencia del estilo [Clásico](#) (es decir, un rectángulo con una etiqueta descriptiva) con la opción de propiedad del menú emergente del estilo [Barra de herramientas](#).

Por defecto, el estilo Bevel tiene un fondo gris claro con una etiqueta en el centro. La apariencia del botón puede ser diferente cuando el cursor pasa por encima de él dependiendo del sistema operativo:

- *Windows* - el botón está resaltado. Cuando utiliza la propiedad "Con menú emergente", aparece un triángulo a la derecha y en el centro del botón.



- *macOS* - el resalte del botón nunca aparece. Cuando utiliza la propiedad "Con menú emergente", aparece un triángulo a la derecha y en la parte inferior del botón.

Ejemplo JSON:

```
"myButton": {  
    "type": "button",  
    "style": "bevel",  
    "text": "OK",  
    "popupPlacement": "linked",  
    "action": "Cancel",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

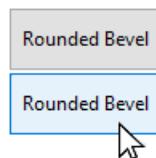
## Bevel redondeado

El estilo de botón Bevel redondeado es casi idéntico al estilo [Bevel](#), excepto que, dependiendo del sistema operativo, las esquinas del botón pueden ser redondeadas. Al igual que el estilo Bevel, el estilo Bevel Redondeado combina la

apariencia del estilo [Clásico](#) con la opción de propiedad del menú emergente del estilo [Barra de herramientas](#).

Por defecto, el estilo Bevel Redondeado tiene un fondo gris claro con una etiqueta en el centro. La apariencia del botón puede ser diferente cuando el cursor pasa por encima de él dependiendo del sistema operativo:

- *Windows* - el botón es idéntico al estilo Bevel. Cuando utiliza la propiedad "Con menú emergente", aparece un triángulo a la derecha y en el centro del botón.



- *macOS* - las esquinas del botón están redondeadas. Cuando utiliza la propiedad "Con menú emergente", aparece un triángulo a la derecha y en la parte inferior del botón.

Ejemplo JSON:

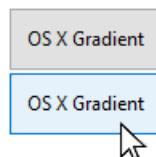
```
"myButton": {  
    "type": "button",  
    "style": "roundedBevel",  
    "text": "OK",  
    "popupPlacement": "none" /  
    "action": "Cancel",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

## OS X Gradient

El estilo del botón OS X Gradient es casi idéntico al estilo [Bevel](#). Al igual que el estilo Bevel, el estilo OS X Gradient combina la apariencia del estilo [Clásico](#) y del estilo [Barra de herramientas](#).

Por defecto, el estilo OS X Gradient tiene un fondo gris claro con una etiqueta en el centro. La apariencia del botón puede ser diferente cuando el cursor pasa por encima de él dependiendo del sistema operativo:

- *Windows* - el botón es idéntico al estilo Bevel. Cuando utiliza la propiedad "Con menú emergente", aparece un triángulo a la derecha del botón.



- *macOS* - el botón se muestra como un botón de dos tonos. Cuando utiliza la propiedad "Con menú emergente", aparece un triángulo a la derecha y en la parte inferior del botón.

Ejemplo JSON:

```

"myButton": {
    "type": "button",
    "style": "gradientBevel",
    "text": "OK",
    "popupPlacement": "linked",
    "action": "Cancel",
    "left": 60,
    "top": 160,
    "width": 100,
    "height": 20
}

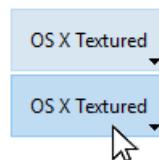
```

## OS X Texturizado

El estilo de botón OS X Textured es casi idéntico al estilo [Bevel](#) pero con un tamaño menor (el tamaño máximo es el de un botón de sistema estándar de macOS). Al igual que el estilo Bevel, el estilo OS X Textured combina la apariencia del estilo [Clásico](#) y del estilo [Barra de herramientas](#).

Por defecto, el estilo OS X Textured aparece como:

- *Windows* - un botón sistema estándar con un fondo gris claro con una etiqueta en el centro. Tiene la particularidad de ser transparente en Vista.



- *macOS* - un botón sistema estándar que muestra un cambio de color de gris claro a gris oscuro. Su altura está predefinida: no es posible ampliarla o reducirla.

Ejemplo JSON:

```

"myButton": {
    "type": "button",
    "style": "texturedBevel",
    "text": "OK",
    "popupPlacement": "separated",
    "action": "Cancel",
    "left": 60,
    "top": 160,
    "width": 100,
    "height": 20
}

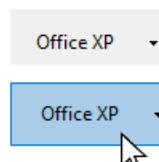
```

## Office XP

El estilo de botón Office XP combina la apariencia del estilo [Clásico](#) y del estilo [Barra de herramientas](#).

Los colores (resaltado y fondo) de un botón con el estilo Office XP se basan en los colores del sistema. La apariencia del botón puede ser diferente cuando el cursor pasa por encima de él dependiendo del sistema operativo:

- *Windows* - su fondo sólo aparece cuando el ratón pasa por encima.



- *macOS* - su fondo se muestra siempre.

Ejemplo JSON:

```
"myButton": {  
    "type": "button",  
    "style": "office",  
    "text": "OK",  
    "popupPlacement": "none"  
    "action": "Cancel",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

## Ayuda

El estilo del botón Ayuda puede utilizarse para mostrar un botón de ayuda estándar del sistema. Por defecto, el estilo Ayuda se muestra como un signo de interrogación dentro de un círculo.



Ejemplo JSON:

```
"myButton": {  
    "type": "button",  
    "style": "help",  
    "text": "OK",  
    "dropping": "custom",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

El estilo Ayuda no soporta las propiedades básicas [Número de estados](#), [ruta de acceso imagen](#) y la posición [Título/Imagen](#).

## Círculo

El estilo de botón Círculo aparece como un botón sistema circular. Este estilo de botón está diseñado para macOS.



En Windows, es idéntico al estilo "Ninguno" (no se tiene en cuenta el círculo del fondo).

Ejemplo JSON:

```
"myButton": {  
    "type": "button",  
    "style": "circular",  
    "text": "OK",  
    "dropping": "custom",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

## Personalizado

El estilo de botón Personalizado acepta una imagen de fondo personalizada y permite gestionar parámetros adicionales como el margen y el desplazamiento del icono.



Ejemplo JSON:

```
"myButton": {  
    "type": "button",  
    "style": "custom",  
    "text": "",  
    "customBackgroundPicture": "/RESOURCES/bknd.png",  
    "icon": "/RESOURCES/custom.png",  
    "textPlacement": "center",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

## Propiedades soportadas

Todos los botones comparten el mismo conjunto de propiedades básicas:

Negrita - Estilo del borde - Abajo - Estilo del botón - Clase - Soltable - Enfocable - Fuente - Color de fuente - Tamaño de fuente - Alto - Mensaje de ayuda - Tam. horizontal - Italic - Izquierda - No representado - Número de estados(1) - Nombre de objeto - Ruta imagen(1) - Derecho - Atajo - Acción estándar - Título - Posición Imagen/Título(1) - Arriba - Tipo - Subrayado - Variable o Expresión - Vertical Sizing - Visibility - Width

(1) No soportado por el estilo Ayuda.

Existen propiedades específicas adicionales, dependiendo del [estilo-de-botón](#):

- [Ruta de acceso fondo](#) - [Margen horizontal](#) - [Desplazamiento icono](#) - [Margen vertical](#) (Personalizado)
- [Botón por defecto](#) (Plano, Clásico)
- [Con menú emergente](#) (Barra de herramientas, Bisel, Bisel redondeado, OS X Gradient, OS X Textured, Office XP, Círculo, Personalizado)

# Rejilla de botones

Una rejilla de botones es un objeto transparente que se coloca sobre una imagen. La imagen debe corresponder a la forma de un array. Cuando se hace clic en uno de los gráficos, éste tendrá un aspecto presionado:

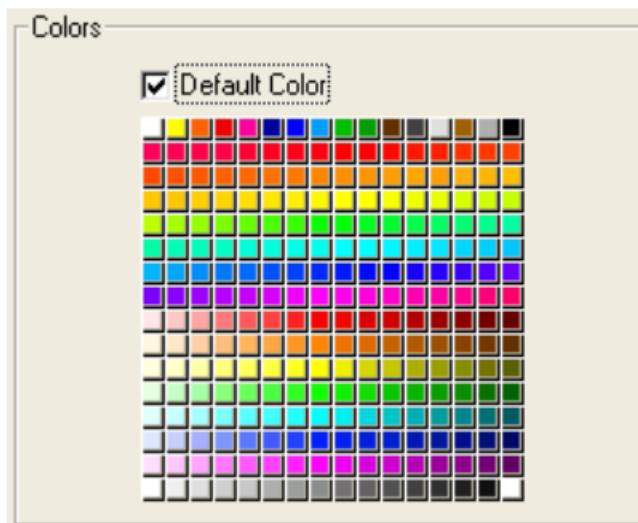


Puede utilizar un objeto rejilla de botones para determinar dónde hace clic el usuario en la imagen. El método objeto utilizaría el evento `On Clicked` y tomaría la acción apropiada dependiendo de la ubicación del clic.

## Crear rejillas de botones

Para crear la rejilla de botones, añada una imagen de fondo al formulario y coloque una rejilla de botones sobre ella. Especifique el número de [líneas](#) y de [columnas](#).

En 4D, se utiliza una rejilla de botones para las paletas de colores:



## Utilizar rejillas de botones

Los botones de la rejilla están numerados de izquierda a derecha y de arriba a abajo. En el ejemplo anterior, la rejilla tiene 16 columnas a lo ancho por 16 líneas a hacia abajo. El botón en la posición superior izquierda devuelve 1 cuando se hace clic. Si se selecciona el botón rojo del extremo derecho de la segunda fila, la rejilla de botones devuelve 32. Si no se selecciona ningún elemento, el valor es 0

## Ir a la página

Puede asociar el `gotoPage` acción estándar a una rejilla de botones. Cuando se selecciona esta acción, 4D mostrará automáticamente la página del formulario que corresponde al número del botón que está seleccionado en la rejilla de botones. Por ejemplo, si el usuario selecciona el décimo botón de la rejilla, 4D mostrará la décima página del formulario actual (si existe).

## Propiedades soportadas

[Estilo del borde](#) - [Inferior](properties\_CoordinatesAndSizing. md#bottom) - [Clase](#) - [Columnas](#) - [Altura](#) - [Consejo de](#)

[ayuda](#) - [Tamaño horizontal](properties\_ResizingOptions. md#horizontal-sizing) - [Izquierda](#) - [Nombre del objeto] (properties\_Object. md#object-name) - [Derecha](#) - [Filas](#) - [Acción estándar](properties\_Action. md#standard-action) - [Superior](#) - [Tipo](properties\_Object. md#type) - [Variable o expresión](#) - [Tamaño vertical](properties\_ResizingOptions. md#vertical-sizing) - [Ancho](#) - [Visibilidad](#)

# Casilla a seleccionar

Una casilla de selección es un tipo de botón utilizado para introducir o mostrar datos binarios (verdadero-falso). Fundamentalmente, está marcado o desmarcado, pero se puede definir un [tercer estado](#).



Las casillas de selección se controlan por métodos o [acciones estándar](#). El método asociado a ella se ejecuta cuando se selecciona la casilla de selección. Como todos los botones, la variable de la casilla de selección se pone en 0 cuando se abre el formulario por primera vez.

Una casilla de selección muestra el texto junto a un pequeño cuadrado. Este texto se define en el área [Título](#) del tema "Objetos" de la Lista de propiedades. Para introducir en esta área un título en forma de referencia XLIFF (ver [Anexo B: arquitectura XLIFF](#)).

## Utilizar casillas de selección

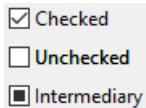
Una casilla de selección puede asociarse a una [variable o expresión](#) de tipo entero o booleano.

- entero: si la casilla está marcada, la variable tiene el valor 1. Cuando no se marca, tiene el valor 0. Si la casilla de selección está en tercer estado (ver más abajo), tiene el valor 2.
- booleano: si la casilla está marcada, la variable tiene el valor `True`. Cuando no se marca, toma el valor `False`.

Una parte o todas las casillas de selección de un formulario pueden estar marcadas o desmarcadas. Las casillas de selección múltiples permiten al usuario seleccionar varias opciones.

## Casilla de selección de tres estados

Los objetos casilla de selección con el [estilo de botón Normal](#) y [Plano](#) aceptan un tercer estado. Este tercer estado es un estado intermedio, que generalmente se utiliza para fines de visualización. Por ejemplo, permite indicar que una propiedad está presente en una selección de objetos, pero no en cada objeto de la selección.



Para activar este tercer estado, debe seleccionar la propiedad [Tres estados](#).

Esta propiedad sólo está disponible para casillas de selección regulares y planas asociadas a [variables o expresiones](#) - las casillas de selección de expresiones booleanas no pueden utilizar la propiedad [Tres estados](#) (una expresión booleana no puede estar en un estado intermedio).

La variable asociada a la casilla de selección devuelve el valor 2 cuando la casilla está en el tercer estado.

En el modo de entrada, las casillas de selección de los tres estados muestran cada estado de forma secuencial, en el siguiente orden: sin marcar / marcado / intermedio / sin marcar, etc. El estado intermedio no suele ser muy útil en el modo de entrada; en el código, basta con forzar el valor de la variable a 0 cuando toma el valor de 2 para pasar directamente del estado comprobado al estado no comprobado.

## Utilizar una acción estándar

Puede asignar una [acción estándar](#) a una casilla de selección para manejar los atributos de las áreas de texto. Por ejemplo, si asigna la acción estándar `fontBold`, en ejecución la casilla de selección gestionará el atributo "negrita" del texto seleccionado en el área actual.

Sólo las acciones que pueden representar un estado verdadero/falso (acciones "marcables") son soportadas por este objeto:

Acciones soportadas	Condiciones de uso (si las hay)
avoidPageBreakInsideEnabled	Área 4D Write Pro únicamente
fontItalic	
fontBold	
fontLinethrough	
fontSubscript	Área 4D Write Pro únicamente
fontSuperscript	Área 4D Write Pro únicamente
fontUnderline	
font/showDialog	Mac únicamente
htmlWYSIWIGEnabled	Área 4D Write Pro únicamente
section/differentFirstPage	Área 4D Write Pro únicamente
section/differentLeftRightPages	Área 4D Write Pro únicamente
spell/autoCorrectionEnabled	
spell/autoDashSubstitutionsEnabled	Mac únicamente
spell/autoLanguageEnabled	Mac únicamente
spell/autoQuoteSubstitutionsEnabled	Mac únicamente
spell/autoSubstitutionsEnabled	
spell/enabled	
spell/grammarEnabled	Mac únicamente
spell/showDialog	Mac únicamente
spell/visibleSubstitutions	
visibleBackground	Área 4D Write Pro únicamente
visibleFooters	Área 4D Write Pro únicamente
visibleHeaders	Área 4D Write Pro únicamente
visibleHiddenChars	Área 4D Write Pro únicamente
visibleHorizontalRuler	Área 4D Write Pro únicamente
visiblePageFrames	Área 4D Write Pro únicamente
visibleReferences	
widowAndOrphanControlEnabled	Área 4D Write Pro únicamente

Para información detallada sobre estas acciones, consulte la sección [Acciones estándar](#).

## Estilos de botones casillas de selección

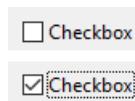
Las casillas de selección utilizan [los estilos de botón](#) para controlar la apariencia general de una casilla de selección, así como sus posibles propiedades. Es posible aplicar diferentes estilos predefinidos a las casillas de selección. Se puede obtener un gran número de variaciones combinando estas propiedades/comportamientos.

Con la excepción de las [propiedades-disponibles](#), muchos objetos casilla de selección son *estructuralmente* idénticos. La diferencia está en el tratamiento de sus variables asociadas.

4D ofrece casillas de selección en los siguientes estilos de botón predefinidos:

### Clásico

El estilo Clásico de botón casilla de selección corresponde a un sistema de casilla de selección estándar (\*es decir, \*, un rectángulo con un título descriptivo):

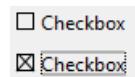


Ejemplo JSON:

```
"myCheckBox": {  
    "type": "checkbox",  
    "style": "regular",  
    "text": "Cancel",  
    "action": "Cancel",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
    "dataSourceTypeHint": "boolean"  
}
```

## Plano

El estilo plano del botón casilla de selección tiene una apariencia minimalista. La naturaleza gráfica del estilo Flat es especialmente útil para los formularios que se van a imprimir.



Ejemplo JSON:

```
"myCheckBox": {  
    "type": "checkbox",  
    "style": "flat",  
    "text": "Cancel",  
    "action": "cancel",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

## Botón barra de herramientas

El estilo del botón barra de herramientas está destinado principalmente a la integración en una barra de herramientas.

El estilo del botón Barra de herramientas tiene un fondo transparente con un título. Suele estar asociado a una [imagen de 4 estados](#).

Ejemplo con estados seleccionado / no seleccionado / subrayado:



Ejemplo JSON:

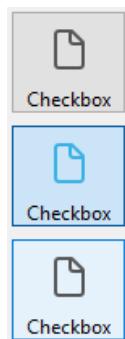
```
"myCheckBox": {  
    "type": "checkbox",  
    "style": "toolbar",  
    "text": "Checkbox",  
    "icon": "/RESOURCES/File.png",  
    "iconFrames": 4  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

## Bevel

El estilo del botón casilla de selección Bevel combina la apariencia del estilo de botón Clásico (es decir, un rectángulo con un título descriptivo) con el comportamiento del botón Barra de herramientas.

El estilo de botón Bevel redondeado tiene un fondo gris claro con un título. Suele estar asociado a una [imagen de 4 estados](#).

Ejemplo con estados seleccionado / no seleccionado / subrayado:



Ejemplo JSON:

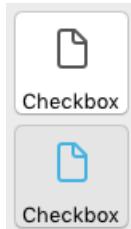
```
"myCheckBox": {  
    "type": "checkbox",  
    "style": "bevel",  
    "text": "Checkbox",  
    "icon": "/RESOURCES/File.png",  
    "iconFrames": 4  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

## Bevel redondeado

El estilo del botón de la casilla de selección Bevel redondeado es casi idéntico al estilo del botón [Bevel](#), excepto que, dependiendo del sistema operativo, las esquinas del botón pueden ser redondeadas. Al igual que el estilo de botón Bevel, el estilo del botón Bevel redondeado combina la apariencia del estilo del botón [Clásico](#) con el comportamiento del estilo del botón [Barra de herramientas](#).

El botón Bevel tiene un fondo gris claro con un título. Suele estar asociado a una [Imagen de 4 estados](#).

Ejemplo en macOS:



En Windows, el estilo de botón Bevel redondeado es idéntico al estilo de botón [Bevel](#).

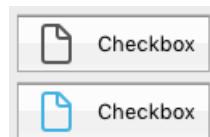
Ejemplo JSON:

```
"myCheckBox": {  
    "type": "checkbox",  
    "style": "roundedBevel",  
    "text": "Checkbox",  
    "icon": "/RESOURCES/File.png",  
    "iconFrames": 4  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

## OS X Gradient

El estilo del botón casilla de selección OS X Gradient es casi idéntico al estilo del botón [Bevel](#). Al igual que el estilo de botón Bevel, el estilo del botón OS X Gradient combina la apariencia del estilo del botón [Clásico](#) con el comportamiento del estilo del botón [Barra de herramientas](#).

El estilo del botón Gradient OS X tiene un fondo gris claro con un título y se puede mostrar como un botón de sistema de dos tonos en macOS. Suele estar asociado a una [Imagen de 4 estados](#).



En Windows, este estilo de botón casilla de selección es idéntico al estilo de botón [Bevel](#).

Ejemplo JSON:

```

"myCheckBox": {
    "type": "checkbox",
    "style": "gradientBevel",
    "text": "Checkbox",
    "icon": "/RESOURCES/File.png",
    "iconFrames": 4
    "left": 60,
    "top": 160,
    "width": 100,
    "height": 20
}

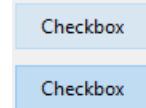
```

## OS X Texturizado

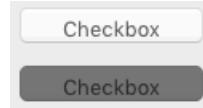
El estilo de botón OS X Textured es similar al estilo del botón [Bevel](#) pero con un tamaño menor (el tamaño máximo es el de un botón de sistema estándar de macOS). Al igual que el estilo de botón Bevel, el estilo del botón OS X Textured combina la apariencia del estilo del botón [Clásico](#) con el comportamiento del estilo del botón [Barra de herramientas](#).

Por defecto, el estilo del botón OS X Textured aparece como:

- *Windows* - un botón sistema estándar con un fondo azul claro con un título en el centro.



- *macOS* - un botón de sistema estándar. Su altura está predefinida: no es posible ampliarla o reducirla.



Ejemplo JSON:

```

"myCheckBox": {
    "type": "checkbox",
    "style": "texturedBevel",
    "text": "Checkbox",
    "left": 60,
    "top": 160,
    "width": 100,
    "height": 20
}

```

## Office XP

El estilo de botón Office XP combina la apariencia del estilo del botón [Clásico](#) con el comportamiento del estilo del [Botón barra de herramientas](#).

Los colores (resaltado y fondo) de una casilla de selección con el estilo de botón Office XP se basan en los colores del sistema. La apariencia de la casilla de selección puede ser diferente cuando el cursor pasa por encima, dependiendo del sistema operativo:

- *Windows* - su fondo sólo aparece cuando el ratón pasa por encima. Ejemplo con estados seleccionado / no seleccionado / subrayado:



- macOS - su fondo se muestra siempre. Ejemplo con estados seleccionado / no seleccionado:



Ejemplo JSON:

```
"myCheckBox": {
    "type": "checkbox",
    "style": "office",
    "text": "Checkbox",
    "action": "fontBold",
    "icon": "/RESOURCES/File.png",
    "iconFrames": 4
    "left": 60,
    "top": 160,
    "width": 100,
    "height": 20
}
```

## Contraer/Desplegar

Este estilo de botón de casilla de selección se puede utilizar para añadir un ícono estándar de contraer/expandir. Estos íconos se utilizan de forma nativa en las listas jerárquicas.

- Windows - el ícono se ve como un [+] o un [-]



- macOS - se ve como un triángulo que apunta hacia la derecha o hacia abajo.



Ejemplo JSON:

```
"myCheckBox": {  
    "type": "checkbox",  
    "style": "disclosure",  
    "method": "mCollapse",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

## Botón de divulgación

En macOS y Windows, una casilla de selección con el estilo de botón "Divulgación" aparece como un botón de información estándar, normalmente utilizado para mostrar/ocultar información adicional. Cuando se utiliza como botón radio, el símbolo del botón apunta hacia abajo con el valor 0 y hacia arriba con el valor 1.

- *Windows*



- *macOS*



Ejemplo JSON:

```
"myCheckBox": {  
    "type": "checkbox",  
    "style": "roundedDisclosure",  
    "method": "mDisclose",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

## Personalizado

El estilo del botón Personalizado acepta una imagen de fondo personalizada y permite gestionar propiedades específicas:

- [Ruta de acceso fondo](#)
- [Desplazamiento icono](#)
- [Horizontal Margin and Margen vertical](#)

Suele estar asociado a una [imagen de 4 estados](#), que puede utilizarse junto con una imagen de fondo [de 4 estados](#).

Ejemplo JSON:

```
"myCheckbox": {  
    "type": "checkbox",  
    "style": "custom",  
    "text": "OK",  
    "icon": "/RESOURCES/smiley.jpg",  
    "iconFrame": 4,  
    "customBackgroundPicture": "/RESOURCES/paper.jpg",  
    "iconOffset": 5, //desplazamiento icono personalizado al hacer clic  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20,  
    "customBorderX": 20,  
    "customBorderY": 5  
}
```

## Propiedades soportadas

Todas las casillas de selección comparten un mismo conjunto de propiedades básicas:

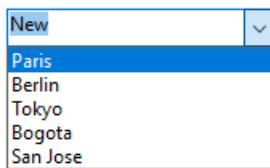
Negrita - Abajo - Estilo de botón - Clase - Editable - Tipo de expresión - Focalizable - Fuente - Color de fuente - Tamaño - Altura - Mensaje de ayuda - Tamaño horizontal - Itálico - Izquierda - Nombre - Derecha - Guardar valor - Ahorro - Acción estándar - Título - Superior - Tipo - Subrayado - Variable o expresión - Tamaño vertical - Visibilidad - Ancho

Existen propiedades específicas adicionales, dependiendo del [estilo-de-botón](#):

- [Ruta de acceso fondo](#) - [Margen horizontal](#) - [Desplazamiento icono](#) - [Margen vertical](#) (Personalizado)
- [Tres estados](#) (Plano, Clásico)
- [Número de estados](#) - [Ruta de acceso imagen](#) - [Posición Título/Imagen](#) (Botón barra de herramientas, Bevel Redondeado, OS X Gradient, OS X Textured, Office XP, Personalizado)

# Combo Box

Un combo box es similar a una [lista desplegable](#), excepto que acepta texto introducido desde el teclado y tiene opciones adicionales.



Fundamentalmente, debe considerar un combo box como un área editable que utiliza su objeto, array o una lista de selección como el conjunto de valores por defecto.

## Gestión de combo boxes

Utilice el evento [On Data Change](#) para gestionar las entradas en el área editable, como lo haría con cualquier objeto del formulario de entrada.

Un combo box se inicializa exactamente igual que una [lista desplegable](#): utilizando un objeto, un array o una lista de selección.

### Utilizar un objeto

Esta funcionalidad sólo está disponible en los proyectos 4D.

Un [objeto](#) encapsulando una [colección](#) puede utilizarse como fuente de datos de un combo box. El objeto debe contener las siguientes propiedades:

Propiedad	Tipo	Descripción
values	Collection	Obligatorio - Colección de valores escalares. Todos los valores deben ser del mismo tipo. Tipos soportados: <ul style="list-style-type: none"><li>• cadenas</li><li>• numbers</li><li>• fechas</li><li>• horas</li></ul> Si está vacío o no está definido, el combo box está vacío
currentValue	igual que Collection	Texto introducido por el usuario

Si el objeto contiene otras propiedades, se ignoran.

Cuando el usuario introduce texto en el combo box, la propiedad `currentValue` del objeto obtiene el texto introducido.

### Utilizar un array

Please refer to [Using an array in the drop-down list page](#) for information about how to initialize the array.

When the user enters text into the combo box, the 0th element of the array gets the entered text.

### Utilizar una lista de selección

If you want to use a combo box to manage the values of an input area (listed field or variable), 4D lets you reference the field or variable directly as the form object's data source. Esto facilita la gestión de los campos/variables listados.

Si utiliza una lista jerárquica, sólo se muestra el primer nivel y se puede seleccionar.

To associate a combo box with a field or variable, you can just enter the name of the field or variable directly in the [Variable or Expression](#) of the form object in the Property List.

When the form is executed, 4D automatically manages the combo box during input or display: when a user chooses a value, it is saved in the field; this field value is shown in the combo box when the form is displayed:

Please refer to Using a choice in the [drop-down list page](#) for more information.

## Opciones

Combo box type objects accept two specific options:

- [Automatic insertion](#): enables automatically adding a value to the data source when a user enters a value that is not found in the list associated with the combo box.
- [Exclusión](#) (lista de valores excluidos): permite establecer una lista cuyos valores no pueden introducirse en el combo box. Si se introduce un valor excluido, no se acepta y se muestra un mensaje de error.

La asociación de una [lista de valores obligatorios](#) no está disponible para los combo box. In an interface, if an object must propose a finite list of required values, then you must use a [drop-down list](#) object.

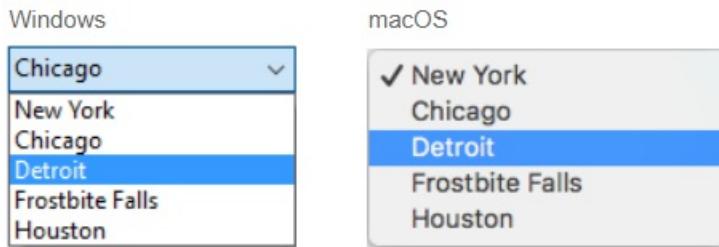
## Propiedades soportadas

[Alpha Format](#) - [Bold](#) - [Bottom](#) - [Choice List](#) - [Class](#) - [Date Format](#) - [Expression Type](#) - [Font](#) - [Font Color](#) - [Font Size](#) - [Height](#) - [Help Tip](#) - [Horizontal Sizing](#) - [Italic](#) - [Left](#) - [Object Name](#) - [Right](#) - [Time Format](#) - [Top](#) - [Type](#) - [Underline](#) - [Variable or Expression](#) - [Vertical Sizing](#) - [Visibility](#) - [Width](#)

# Lista desplegable

Las listas desplegables son objetos de formulario que permiten al usuario seleccionar de una lista. Los elementos mostrados en la lista desplegable se gestionan mediante un objeto, array, una lista de selección o una acción estándar.

En macOS, las listas desplegables también se denominan a veces "menú emergente". Ambos nombres se refieren a los mismos objetos. Como muestra el siguiente ejemplo, el aspecto de estos objetos puede variar ligeramente según la plataforma:



## Tipos de listas desplegables

You can create different types of drop-down lists with different features. To define a type, select the appropriate Expression Type and Data Type values in the Property list, or use their JSON equivalent.

Tipo	Funcionalidades	Tipo de expresión	Tipos de datos	Definición JSON
Objeto	Built upon a collection	Objeto	Numeric, Text, Date o Time	<code>dataSourceTypeHint: object + numberFormat: &lt;format&gt; or textFormat: &lt;format&gt; or dateFormat: &lt;format&gt; or timeFormat: &lt;format&gt;</code>
Array	Basado en un array	Array	Numeric, Text, Date o Time	<code>dataSourceTypeHint: arrayNumber or arrayText or arrayDate or arrayTime</code>
Lista de selección guardada como valor	Built upon a choice list (standard)	Lista	Valor del elemento seleccionado	<code>dataSourceTypeHint: text + saveAs: value</code>
Choice list saved as reference	Built upon a choice list. La posición del elemento es guardada	Lista	Referencia del elemento seleccionado	<code>dataSourceTypeHint: integer + saveAs: reference</code>
Lista de selección jerárquica	Puede mostrar contenido jerárquico	Lista	List reference	<code>dataSourceTypeHint: integer</code>
Acción estándar	Creado automáticamente por la acción	any	<i>toda referencia de lista excepto</i>	<code>any definition + action: &lt;action&gt; (+ focusable: false for actions applying to other areas)</code>

## Gestión de listas desplegables

### Utilizar un objeto

Esta funcionalidad sólo está disponible en proyectos 4D.

Un [objeto](#) encapsulando una [colección](#) puede utilizarse como fuente de datos de una lista desplegable. El objeto debe contener las siguientes propiedades:

Propiedad	Tipo	Descripción
values	Collection	Obligatorio - Colección de valores escalares. Todos los valores deben ser del mismo tipo. Tipos soportados: <ul style="list-style-type: none"><li>• strings</li><li>• numbers</li><li>• fechas</li><li>• horas</li></ul> Si está vacío o no está definido, la lista desplegable está vacía
index	number	Index of the currently selected item (value between 0 and <code>collection.length-1</code> ). If you set -1, <code>currentValue</code> is displayed as a placeholder string
currentValue	igual que Collection	Currently selected item (used as placeholder value if set by code)

Si el objeto contiene otras propiedades, se ignoran.

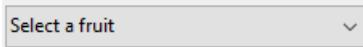
To initialize the object associated to the drop-down list, you can:

- Introduzca una lista de valores por defecto en las propiedades del objeto seleccionando "<Static List>" en el tema [Fuente de datos](#) de la lista de propiedades. The default values are loaded into an object automatically.
- Execute code that creates the object and its properties. For example, if "myList" is the [variable](#) associated to the drop-down list, you can write in the [On Load](#) form event:

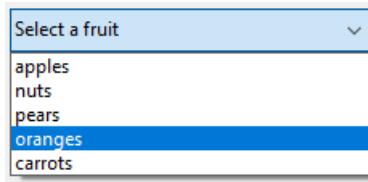
```
// Form.myDrop is the datasource of the form object

Form.myDrop:=New object
Form.myDrop.values:=New collection("apples"; "nuts"; "pears"; "oranges"; "carrots")
Form.myDrop.index:=-1 //currentValue is a placeholder
Form.myDrop.currentValue:="Select a fruit"
```

The drop-down list is displayed with the placeholder string:



After the user selects a value:



```
Form.myDrop.values // ["apples", "nuts", "pears", "oranges", "carrots"]
Form.myDrop.currentValue // "oranges"
Form.myDrop.index // 3
```

## Utilizar un array

Un [array](#) es una lista de valores en memoria a la que se hace referencia por el nombre del array. Una lista desplegable puede mostrar un array como una lista de valores cuando se hace clic en ella.

Para inicializar el array asociado a la lista desplegable, puede:

- Introduzca una lista de valores por defecto en las propiedades del objeto seleccionando "<Static List>" en el tema [Fuente de datos](#) de la lista de propiedades. Los valores por defecto se cargan en un array automáticamente. Puede referirse al array utilizando el nombre de la variable asociada al objeto.
- Antes de mostrar el objeto, ejecute el código que asigna valores a los elementos del array. Por ejemplo:

```
ARRAY TEXT(aCities;6)
aCities{1}:= "Philadelphia"
aCities{2}:= "Pittsburg"
aCities{3}:= "Grand Blanc"
aCities{4}:= "Bad Axe"
aCities{5}:= "Frostbite Falls"
aCities{6}:= "Green Bay"
```

En este caso, el nombre de la [variable](#) asociada al objeto en el formulario debe ser `aCities`. Este código podría colocarse en el método formulario y ejecutarse cuando se ejecute el evento formulario `On Load`.

- Antes de que se muestre el objeto, cargue los valores de una lista en el array utilizando el comando [LIST TO ARRAY](#). Por ejemplo:

```
LIST TO ARRAY("Cities";aCities)
```

En este caso también, el nombre de la [variable](#) asociada al objeto en el formulario debe ser `aCities`. Este código puede ejecutarse en lugar de las sentencias de asignación mostradas anteriormente.

Si necesita guardar la elección del usuario en un campo, deberá utilizar una sentencia de asignación que se ejecute después de aceptar el registro. El código podría ser así:

```
Case of
:(Form event=On Load)
  LIST TO ARRAY("Cities";aCities)
  If(Record number([People])<0) `nuevo registro
    aCities:=3 `mostrar un valor por defecto
  Else `registro existente, mostrar valor almacenado
    aCities:=Find in array(aCities;City)
  End if
:(Form event=On Clicked) `el usuario modifica la selección
  City:=aCities{aCities} `el campo obtiene un nuevo valor
:(Form event=On Validate)
  City:=aCities{aCities}
:(Form event=On Unload)
  CLEAR VARIABLE(aCities)
End case
```

Debe seleccionar cada evento que prueba en sus sentencia Case. Los arrays siempre contienen un número finito de elementos. La lista de elementos es dinámica y puede ser modificada por un método. Los elementos de un array pueden modificarse, ordenarse y añadirse.

## Utilizar una lista de selección

If you want to use a drop-down list to manage the values of an input area (listed field or variable), 4D lets you reference the field or variable directly as the drop-down list's [data source](#). Esto facilita la gestión de los campos/variables listados.

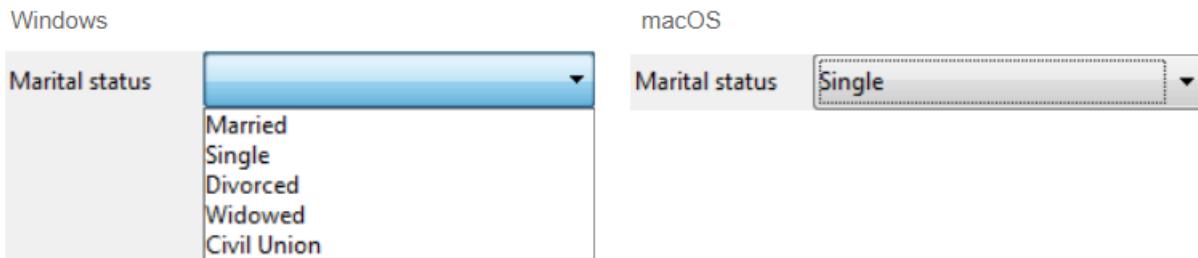
For example, in the case of a "Color" field that can only contain the values "White", "Blue", "Green" or "Red", it is possible to create a list containing these values and associate it with a drop-down list that references the 4D "Color" field. 4D se encarga entonces de gestionar automáticamente la entrada y la visualización del valor actual en el formulario.

Si utiliza una lista jerárquica, sólo se muestra el primer nivel y se puede seleccionar. Si utiliza una lista jerárquica, sólo se muestra el primer nivel y se puede seleccionar.

To associate a drop-down list with a field or variable, enter the name of the field or variable directly as the [Variable or Expression](#) field of the drop-down list in the Property List.

It is not possible to use this feature with an object or an array drop-down list. If you enter a field name in the "Variable or Expression" area, then you must use a choice list.

When the form is executed, 4D automatically manages the drop-down list during input or display: when a user chooses a value, it is saved in the field; this field value is shown in the drop-down list when the form is displayed:

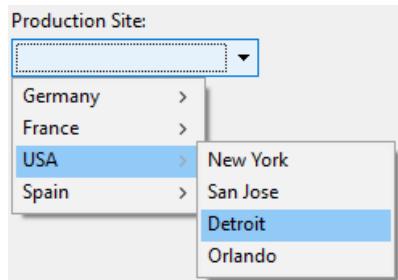


#### Valor del elemento seleccionado o Referencia del elemento seleccionado

When you have associated a drop-down list with a choice list and with a field or a variable, you can set the [Data Type](#) property to Selected item value or Selected item reference. Esta opción permite optimizar el tamaño de los datos guardados.

#### Uso de una lista de selección jerárquica

A hierarchical drop-down list has a sublist associated with each item in the list. Here is an example of a hierarchical drop-down list:



In forms, hierarchical drop-down lists are limited to two levels.

You can assign the hierarchical choice list to the drop-down list object using the [Choice List](#) field of the Property List.

You manage hierarchical drop-down lists using the Hierarchical Lists commands of the 4D Language. All commands that support the `(*; "name")` syntax can be used with hierarchical drop-down lists, e.g. [List item parent](#).

#### Utilizar una acción estándar

You can build automatically a drop-down list using a [standard action](#). This feature is supported in the following contexts:

- Uso de la acción estándar `gotoPage`. In this case, 4D will automatically display the [page of the form](#) that corresponds to the number of the item that is selected. For example, if the user selects the 3rd item, 4D will display the third page of the current form (if it exists). At runtime, by default the drop-down list displays the page numbers (1, 2...).
- Use of a standard action that displays a sublist of items, for example `backgroundColor`. This feature requires that:

- a styled text area ([4D Write Pro area](#) or [input](#) with [multistyle](#) property) is present in the form as the standard action target.
- the [focusable](#) property is not set to the drop-down list. At runtime the drop-down list will display an automatic list of values, e.g. background colors. You can override this automatic list by assigning in addition a choice list in which each item has been assigned a custom standard action.

This feature cannot be used with a hierarchical drop-down list.

## Propiedades soportadas

[Alpha Format](#) - [Bold](#) - [Bottom](#) - [Button Style](#) - [Choice List](#) - [Class](#) - [Data Type \(expression type\)](#) - [Data Type \(list\)](#) - [Date Format](#) - [Expression Type](#) - [Focusable](#) - [Font](#) - [Font Color](#) - [Font Size](#) - [Height](#) - [Help Tip](#) - [Horizontal Sizing](#) - [Italic](#) - [Left](#) - [Not rendered](#) - [Object Name](#) - [Right](#) - [Standard action](#) - [Save value](#) - [Time Format](#) - [Top](#) - [Type](#) - [Underline](#) - [Variable or Expression](#) - [Vertical Sizing](#) - [Visibility](#) - [Width](#)

# Área de grupo

Un área de grupo es un objeto estático que permite ensamblar visualmente varios objetos de formulario:

The screenshot shows a user interface element labeled "Employee Info". Inside this box, there are three input fields: "First name" with the value "[Employee]firstName", "Last name" with the value "[Employee]lastName", and "Salary" with the value "[Employee]salary".

El nombre de un área de grupo es un texto estático; puede utilizar una referencia "localizable" como con cualquier etiqueta de 4D (ver [Utilizar las referencias en los textos estáticos](#) y la sección *Arquitectura XLIFF* en el manual Diseño de 4D.

Ejemplo JSON:

```
"myGroup": {  
    "type": "groupBox",  
    "title": "Employee Info"  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

Propiedades soportadas

[Inferior](#) - [CSS Class](#) - [Fuente](#) - [Color de la fuente](#) - [Tamaño de la fuente](#) - [Altura](#) - [Alineamiento horizontal](#) - [Dim. horizontal](#) - [Itálica](#) - [Izquierda](#) - [Nombre del objeto](#) - [Derecha](#) - [Título](#) - [Arriba](#) - [Tipo](#) - [Subrayado](#) - [Dim. vertical](#) - [Visibilidad](#) - [Ancho](#)

# Entrada

Las entradas le permiten añadir expresiones editables o no editables como [campos](#) y [variables](#) de base de datos a sus formularios. Las entradas pueden manejar datos basados en caracteres (texto, fechas, números...) o imágenes:



Las entradas pueden contener [expresiones asignables o no asignables](#).

Además, las entradas pueden ser [editables o no editables](#). Una entrada introducible acepta los datos. Puede definir los controles de entrada de datos para el objeto. Una entrada no editable sólo puede mostrar valores, pero no puede ser editada por el usuario.

Puedes gestionar los datos con los [métodos](#) objeto o formulario.

## Ejemplo JSON:

```
"miTexto": {  
    "tipo": "input", //define el tipo de objeto  
    "spellcheck": true, //activa la verificación ortográfica  
    "left": 60, //posición izquierda en el formulario  
    "top": 160, //posición superior en el formulario  
    "width": 100, //ancho del objeto  
    "height": 20 //altura del objeto  
}
```

## Propiedades soportadas

Permitir selector de fuente/color - Formato Alfa - Comprobación ortográfica automática - Negrita - Probar cuando False/Text cuando True - Estilo de línea de borde - Abajo - Lista de opciones - Clase - Menú de contexto - Formato de fecha - Valor por defecto - Arrastrable - Soltable - Editable - Filtro de entrada - Lista de excluidos - Tipo de expresión - Color de relleno - Fuente - Color de fuente - Tamaño de fuente - Altura - Ocultar rectángulo de enfoque - Alineación horizontal - Barra de desplazamiento horizontal - Tamaño horizontal - Itálica - Izquierda - Multilínea - Multiestilo - Formato numérico - Nombre del objeto - Orientación - Formato de imagen - Titular - Marco de impresión - Lista requerida - Derecha - Selección siempre visible - Almacenar con etiquetas de estilo por defecto - Texto cuando False/Text cuando True - Formato de tiempo - Arriba - Tipo - Subrayado - Variable o expresión - Barra de desplazamiento vertical - Tamaño vertical - Visibilidad - Ancho - Ajuste de texto

## Alternativas

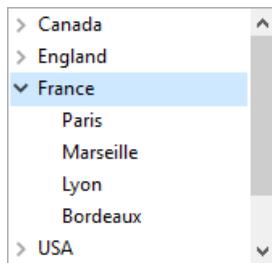
También puede representar expresiones de campos y de variables en sus formularios utilizando objetos alternativos, más concretamente:

- Puede mostrar e introducir datos de los campos de la base directamente en las columnas [de tipo List box](#).
- Puede representar un campo de lista o una variable directamente en un formulario utilizando los objetos [Menús desplegables/Listas desplegables](#) y [Combo Box](#).
- Puede representar una expresión booleana como una [casilla de selección](#) o como un objeto [botón radio](#).



# Lista jerárquica

Las listas jerárquicas son objetos formulario que pueden utilizarse para mostrar datos en forma de listas con uno o más niveles que pueden desplegarse o contraerse.



Cuando corresponda, el ícono desplegar/contraer se mostrará automáticamente a la izquierda del elemento. Las listas jerárquicas soportan un número ilimitado de subniveles.

## Fuente de datos de lista jerárquica

El contenido de un objeto formulario lista jerárquica se puede inicializar de una de las siguientes maneras:

- Asociar una [lista de opciones](#) existente al objeto. La lista de elección debe haber sido definida en el editor de listas en modo Diseño.
- Asigne directamente una referencia de lista jerárquica a la [variable o expresión](#) asociada al objeto formulario.

En ambos casos, se gestiona una lista jerárquica en tiempo de ejecución a través de su referencia `ListRef`, utilizando los comandos [lista jerárquica](#) del lenguaje 4D.

## RefList y nombre de objeto

Una lista jerárquica es a la vez un objeto de lenguaje existente en memoria y un objeto de formulario.

El objeto de lenguaje está referenciado por un ID interno único de tipo Entero largo, designado por `ListRef` en el manual de Lenguaje 4D. Este ID es devuelto por los comandos que se pueden usar para crear listas: `New list`, `Copy list`, `Load list`, `BLOB to list`. Sólo hay una instancia del objeto lenguaje en la memoria y cualquier modificación realizada en este objeto se traslada inmediatamente a todos los lugares donde se utiliza.

El objeto de formulario no es necesariamente único: puede haber varias representaciones de la misma lista jerárquica en el mismo formulario o en otros diferentes. Al igual que con otros objetos formulario, se especifica el objeto en el lenguaje utilizando la sintaxis (\*; "NomLista", etc.).

Conecte el "objeto lenguaje" lista jerárquica con el "objeto de formulario" lista jerárquica por medio de la variable que contiene el valor RefList. Por ejemplo, si has asociado la [variable](#) myList al objeto de formulario, puede escribir:

```
mylist:=New list
```

Cada representación de la lista tiene sus propias características específicas y comparte características comunes con todas las demás representaciones. Las siguientes características son específicas de cada representación de la lista:

- La selección,
- El estado desplegado/colapsado de sus elementos,
- La posición del cursor de desplazamiento.

Las demás características (fuente, tamaño de fuente, estilo, control de entrada, color, contenido de la lista, iconos, etc.) son comunes a todas las representaciones y no pueden modificarse por separado. Por consiguiente, cuando se utilizan comandos basados en la configuración expandida/colapsada o en el elemento actual, por ejemplo `Count list items` (cuando no se pasa el parámetro final \*), es importante poder especificar la representación que se utilizará sin ninguna ambigüedad.

Debe utilizar el identificador `RefLista` con los comandos del lenguaje cuando quiera especificar la lista jerárquica que se encuentra en la memoria. En cambio, si desea especificar la representación al nivel del formulario de un objeto Lista jerárquica, debe utilizar el nombre del objeto (tipo cadena) en el comando, mediante la sintaxis estándar (\*; "NomLista", etc.).

En el caso de los comandos que definen propiedades, la sintaxis basada en el nombre del objeto no significa que sólo el objeto formulario especificado será modificado por el comando, sino que la acción del comando se basará en el estado de este objeto. Las características comunes de las listas jerárquicas se modifican siempre en todas sus representaciones. Por ejemplo, si pasa la instrucción:

```
SET LIST ITEM FONT(*;"mylist1";*;thefont)
```

... está indicando que quiere modificar la fuente de un elemento de la lista jerárquica asociada al objeto de formulario *mylist1*. El comando tendrá en cuenta el elemento actual del objeto *mylist1* para definir el elemento a modificar, pero esta modificación se trasladará a todas las representaciones de la lista en todos los procesos.

## Soporte de @

Al igual que con otros comandos de gestión de propiedades de objetos, es posible utilizar el carácter "@" en el parámetro `NomLista`. Por regla general, esta sintaxis se utiliza para designar un conjunto de objetos del formulario. Sin embargo, en el contexto de los comandos de listas jerárquicas, esto no se aplica en todos los casos. Esta sintaxis tendrá dos efectos diferentes según el tipo de comando:

- Para los comandos que fijan propiedades, esta sintaxis designa todos los objetos cuyo nombre corresponde (comportamiento estándar). Por ejemplo, el parámetro "LH@" designa todos los objetos del tipo lista jerárquica cuyo nombre empieza por "LH."
  - `DELETE FROM LIST`
  - `INSERT IN LIST`
  - `SELECT LIST ITEMS BY POSITION`
  - `SET LIST ITEM`
  - `SET LIST ITEM FONT`
  - `SET LIST ITEM ICON`
  - `SET LIST ITEM PARAMETER`
  - `SET LIST ITEM PROPERTIES`
- Para los comandos que recuperan propiedades, esta sintaxis designa el primer objeto cuyo nombre corresponde:
  - `Count list items`
  - `Find in list`
  - `GET LIST ITEM`
  - `Get list item font`
  - `GET LIST ITEM ICON`
  - `GET LIST ITEM PARAMETER`
  - `GET LIST ITEM PROPERTIES`
  - `List item parent`
  - `List item position`
  - `Selected list items`

## Comandos genéricos utilizables con listas jerárquicas

Es posible modificar la apariencia de una lista jerárquica en un formulario utilizando varios comandos 4D genéricos. Puede pasar a estos comandos el nombre del objeto de la lista jerárquica (utilizando el parámetro \*), o su nombre de variable (que contiene el valor `ListRef`):

- `OBJECT SET FONT`
- `OBJECT SET FONT STYLE`
- `OBJECT SET FONT SIZE`
- `OBJECT SET COLOR`
- `OBJECT SET FILTER`
- `OBJECT SET ENTERABLE`
- `OBJECT SET SCROLLBAR`
- `OBJECT SET SCROLL POSITION`
- `OBJECT SET RGB COLORS`

Recordatorio: excepto `OBJECT SET SCROLL POSITION`, estos comandos modifican todas las representaciones de una misma lista, aunque sólo se especifique una lista a través de su nombre de objeto.

## Prioridad de los comandos de propiedad

Ciertas propiedades de las listas jerárquicas (por ejemplo, el atributo `editable` o el color) pueden definirse de diferentes maneras: en las propiedades del formulario, mediante un comando del tema "Propiedades de los objetos" o mediante un comando del tema "Lista jerárquica". Cuando se utilizan los tres medios para definir las propiedades de la lista, se aplica el siguiente orden de prioridad:

1. Comandos del tema "Lista jerárquica"
2. Comandos genéricos de propiedad de objeto
3. Propiedad formulario

Este principio se aplica independientemente del orden de llamada de los comandos. Si una propiedad de un elemento se modifica individualmente a través de un comando de lista jerárquica, el comando de propiedad de objeto equivalente no tendrá ningún efecto sobre este elemento, incluso si se llama posteriormente. Por ejemplo, si el color de un elemento se modifica a través del comando `SET LIST ITEM PROPERTIES`, el comando `OBJECT SET COLOR` no tendrá ningún efecto sobre este elemento.

## Gestión de los elementos por posición o por referencia

Normalmente se puede trabajar de dos maneras con el contenido de las listas jerárquicas: por posición o por referencia.

- Cuando se trabaja por posición, 4D se basa en la posición con respecto a los elementos de la lista que aparecen en pantalla para identificarlos. El resultado será diferente según se expandan o colapsen determinados elementos jerárquicos. Tenga en cuenta que en el caso de las representaciones múltiples, cada objeto formulario tiene su propia configuración de elementos expandidos/colapsados.
- Cuando se trabaja por referencia, 4D se basa en el número de identificación `itemRef` de los elementos de la lista. Así, cada elemento puede especificarse individualmente, independientemente de su posición o de su visualización en la lista jerárquica.

### Utilizar los números de referencia de los artículos (`itemRef`)

Cada elemento de una lista jerárquica tiene un número de referencia (`itemRef`) del tipo Entero largo. Este valor sólo está destinado a su propio uso: 4D simplemente lo mantiene.

Atención: puede utilizar cualquier tipo de valor entero largo como número de referencia, excepto 0. De hecho, para la mayoría de los comandos de este tema, se utiliza el valor 0 para especificar el último elemento añadido a la lista.

He aquí algunos consejos para utilizar los números de referencia:

1. No es necesario identificar cada elemento con un número único (nivel principiante).
  - Primer ejemplo: se construye por programación un sistema de pestañas, por ejemplo, una libreta de direcciones. Como el sistema devuelve el número de la pestaña seleccionada, probablemente no necesitará más

información que ésta. En este caso, no se preocupe por los números de referencia de los elementos: pase un valor cualquiera (excepto 0) en el parámetro `itemRef`. Tenga en cuenta que para un sistema de libreta de direcciones, puede predefinir una lista A, B, ..., Z en el modo Diseño. También se puede crear por programación para eliminar las letras para las que no hay registros.

- Segundo ejemplo: al trabajar con una base, se construye progresivamente una lista de palabras clave. Puede guardar esta lista al final de cada sesión utilizando los comandos `SAVE LIST` o `LIST TO BLOB` y volver a cargarla al comienzo de cada nueva sesión utilizando el `Load list` o `BLOB to list`. Puede mostrar esta lista en una paleta flotante; cuando cada usuario hace clic en una palabra clave de la lista, el elemento elegido se inserta en el área introducible que está seleccionada en el proceso en primer plano. Lo importante es que sólo procese el elemento seleccionado, porque el comando `Select list items` devuelve la posición del elemento que debe procesar. Cuando se utiliza este valor de posición, se obtiene el título del elemento mediante el comando `GET LIST ITEM`. También en este caso, no es necesario identificar cada elemento individualmente; puede pasar cualquier valor (excepto 0) en el parámetro `itemRef`.
2. Es necesario identificar parcialmente los elementos de la lista (nivel intermedio).  
El número de referencia del elemento se utiliza para almacenar la información necesaria cuando se debe trabajar con el elemento; este punto se detalla en el ejemplo del comando `APPEND TO LIST`. En este ejemplo, utilizamos los números de referencia de los artículos para almacenar los números de registro. Sin embargo, debemos ser capaces de establecer una distinción entre los elementos que corresponden a los registros [Department] y los que corresponden a los registros [Employees].
3. Identifique todos los elementos de la lista individualmente (nivel avanzado).  
Programe una gestión elaborada de listas jerárquicas en la que es absolutamente necesario poder identificar cada elemento individualmente en cada nivel de la lista. Una forma sencilla de ponerlo en práctica es mantener un contador personal. Suponga que crea una lista `hList` utilizando el comando `APPEND TO LIST`. En esta etapa, se inicializa un contador `vhlCounter` en 1. Cada vez que se llama a `APPEND TO LIST` o `INSERT IN LIST`, se incrementa este contador (`vhlCounter:=vhlCounter+1`), y se pasa el número del contador como número de referencia del elemento. El truco consiste en no disminuir nunca el contador cuando se eliminan elementos: el contador sólo puede aumentar. De este modo, se garantiza la unicidad de los números de referencia de los elementos. Como estos números son de tipo Entero largo, puede añadir o insertar más de dos mil millones de elementos en una lista que ha sido reiniciada... (sin embargo, si está trabajando con un número tan grande de elementos, esto suele significar que debes utilizar una tabla en lugar de una lista)

Si se utilizan operadores Bitwise, también se pueden utilizar los números de referencia de los elementos para almacenar información que se puede poner en un Entero largo, es decir, 2 enteros, valores de 4 bytes o de nuevo 32 booleanos.

## ¿Cuándo necesita números de referencia únicos?

En la mayoría de los casos, cuando se utilizan listas jerárquicas con fines de interfaz de usuario y cuando sólo se trata del elemento seleccionado (por un clic o arrastrado), no será necesario utilizar los números de referencia de los elementos en absoluto. Con `Selected list items` y `GET LIST ITEM`, tiene todo lo que necesita para tratar con el elemento seleccionado actualmente. Además, comandos como `INSERT IN LIST` y `DELETE FROM LIST` permiten manipular la lista "relativamente" con respecto al elemento seleccionado.

Básicamente, es necesario tratar con los números de referencia de los elementos cuando se quiere acceder directamente a cualquier elemento de la lista de forma programada y no necesariamente al actualmente seleccionado en la lista.

## Elemento modificable

Puede controlar si los elementos de la lista jerárquica pueden ser modificados por el usuario utilizando el atajo de teclado Alt+clic(Windows) / Opción+clic (macOS), o realizando una pulsación larga sobre el texto del elemento.

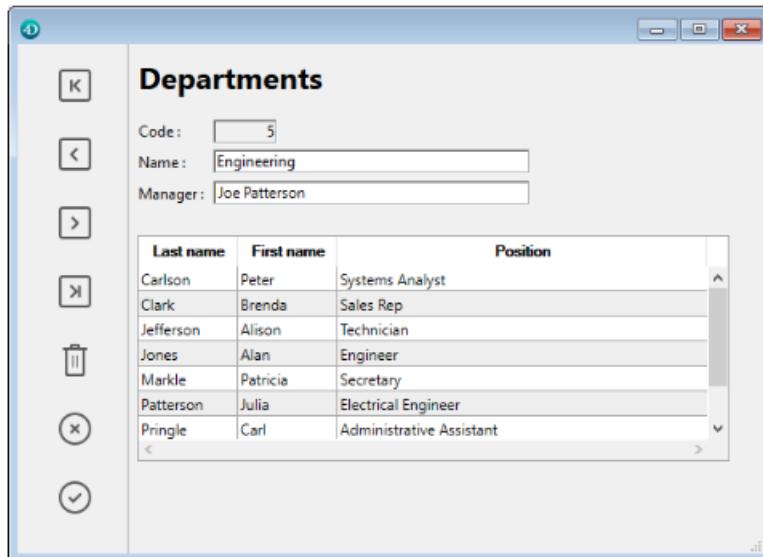
- Sea cual sea la fuente de datos de la lista jerárquica, puede controlar todo el objeto con la propiedad `Editable`.
- Además, si llena la lista jerárquica utilizando una lista creada en el editor de listas, puede controlar si un elemento de una lista jerárquica es modificable mediante la opción Elemento modificable del editor de listas. Para más información, consulte [Definir las propiedades de la lista](#).

## Propiedades soportadas

Negrita - Estilo de línea de borde - Abajo - Lista de opciones - Clase - Arrastrable - Soltable - Editable - Filtro de entrada - Color de relleno - Enfocable - Fuente - Color de fuente - Tamaño de fuente - Altura - Mensaje de ayuda - Ocultar rectángulo de enfoque - Barra de desplazamiento horizontal - Tamaño horizontal - Itálica - Izquierda - Multiseleccionable - Nombre del objeto - Derecha - Arriba - Tipo - Subrayado - Barra de desplazamiento vertical - Tamaño vertical - Variable o expresión - Visibilidad - Ancho

# List Box

Los list boxes son objetos activos complejos que permiten mostrar e introducir datos en forma de columnas sincronizadas. Pueden vincularse a contenidos de la base de datos, como selecciones de entidades y secciones de registros, o a cualquier contenido del lenguaje, como colecciones y arrays. Incluyen funciones avanzadas relativas a la entrada de datos, la ordenación de columnas, la gestión de eventos, el aspecto personalizado, el desplazamiento de columnas, etc.



Un list box contiene una o varias columnas cuyo contenido se sincroniza automáticamente. El número de columnas es, en teoría, ilimitado (depende de los recursos de la máquina).

## Generalidades

### Principios de utilización básicos

Durante la ejecución, los list box permiten visualizar e introducir datos en forma de listas. Para hacer que una celda sea editable ([si se permite la entrada para la columna](#)), basta con pulsar dos veces sobre el valor que contiene:

Last name	First name
James	Henry
Jameson	Marc

Los usuarios pueden introducir y mostrar el texto en varias líneas dentro de una celda de list box. Para añadir un salto de línea, presione Ctrl+Retorno de carro en Windows o Opción+Retorno de carro en macOS.

En las celdas se pueden mostrar booleanos e imágenes, así como fechas, horas o números. Es posible ordenar los valores de las columnas haciendo clic en un encabezado ([ordenación estándar](#)). Todas las columnas se sincronizan automáticamente.

También es posible cambiar el tamaño de cada columna, y el usuario puede modificar el orden de las [columnas](#) y [líneas](#) moviéndolas con el ratón, si esta acción está autorizada. Tenga en cuenta que los list box se pueden utilizar en [modo jerárquico](#).

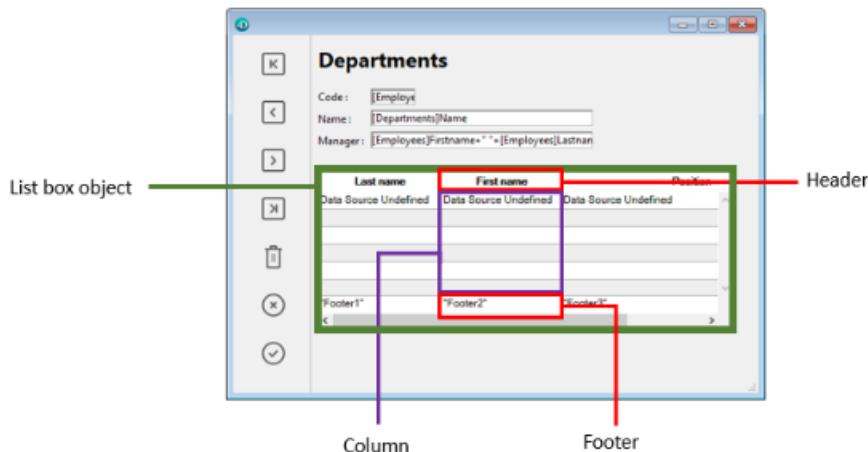
El usuario puede seleccionar una o varias líneas utilizando los atajos estándar: Mayúsculas+clic para una selección adyacente y Ctrl+clic (Windows) o Comando+clic (macOS) para una selección no adyacente.

## Partes de list box

Un list box se compone de cuatro partes distintas:

- el objeto list box en su totalidad,

- las columnas,
- los encabezados de las columnas, y
- los pies de las columnas.



Cada parte tiene su propio nombre y propiedades específicas. Por ejemplo, el número de columnas o el color alternativo de cada línea se define en las propiedades del objeto list box, el ancho de cada columna se define en las propiedades de las columnas y el tipo de fuente del encabezado se define en las propiedades de los encabezados.

Es posible añadir un método objeto al objeto list box y/o a cada columna del list box. Los métodos objeto se llaman en el siguiente orden:

1. Método objeto de cada columna
2. Método objeto del list box

El método objeto de columna obtiene los eventos que se producen en su [encabezado](#) y [pie](#).

## Tipos de list box

Hay varios tipos de list box, con sus propios comportamientos y propiedades específicas. El tipo de list box depende de su [propiedad Fuente de datos](#):

- Arrays: cada columna está ligada a un array 4D. Los list boxes basados en arrays pueden mostrarse como [cajas de lista jerárquicas](#).
- Selección (Selección actual o Selección con nombre): cada columna está vinculada a una expresión (por ejemplo, un campo) que se evalúa para cada registro de la selección.
- Collection or Entity selection : each column is bound to an expression which is evaluated for every element of the collection or every entity of the entity selection.

**Nota:** It is not possible to combine different list box types in the same list box object. The data source is set when the list box is created. It is then no longer possible to modify it by programming.

## Gestión de list boxes

You can completely configure a list box object through its properties, and you can also manage it dynamically through programming.

The 4D Language includes a dedicated "List Box" theme for list box commands, but commands from various other themes, such as "Object properties" commands or `EDIT ITEM`, `Displayed line number` commands can also be used. Refer to the [List Box Commands Summary](#) page of the *4D Language* reference for more information.

# Objetos tipo List box

## List box de tipo array

En un list box de tipo array, cada columna debe estar asociada a un array unidimensional 4D; se pueden utilizar todos los tipos de array, a excepción de los arrays de punteros. The number of rows is based on the number of array elements.

By default, 4D assigns the name "ColumnX" to each column. You can change it, as well as other column properties, in the [column properties](#). The display format for each column can also be defined using the [OBJECT SET FORMAT](#) command.

Array type list boxes can be displayed in [hierarchical mode](#), with specific mechanisms.

Con los list box de tipo array, los valores introducidos o mostrados se gestionan utilizando el lenguaje 4D. You can also associate a [choice list](#) with a column in order to control data entry. Los valores de las columnas se gestionan mediante comandos de alto nivel del tema List box (como [LISTBOX INSERT ROWS](#) o [LISTBOX DELETE ROWS](#)), así como comandos de manipulación de arrays. Por ejemplo, para inicializar el contenido de una columna, puede utilizar la siguiente instrucción:

```
ARRAY TEXT(varCol;size)
```

También puede utilizar una lista:

```
LIST TO ARRAY("ListName";varCol)
```

Atención: cuando un objeto List box contiene varias columnas de diferentes tamaños, sólo se mostrará el número de elementos del array (columna) más pequeño. Debe asegurarse de que cada array tenga el mismo número de elementos que los demás. Además, si una columna del list box está vacía (esto ocurre cuando el array asociado no fue declarado o dimensionado correctamente con el lenguaje), el list box no muestra nada.

## List box de tipo selección

En este tipo de list box, cada columna puede estar asociada a un campo (por ejemplo [\[Employees\]LastName](#)) o a una expresión. The expression can be based on one or more fields (for example, [\[Employees\]FirstName+" "](#) [\[Employees\]LastName](#)) or it may simply be a formula (for example [String\(Milliseconds\)](#)). The expression can also be a project method, a variable or an array item. You can use the [LISTBOX SET COLUMN FORMULA](#) and [LISTBOX INSERT COLUMN FORMULA](#) commands to modify columns programmatically.

The contents of each row is then evaluated according to a selection of records: the current selection of a table or a named selection.

In the case of a list box based on the current selection of a table, any modification done from the database side is automatically reflected in the list box, and vice versa. The current selection is therefore always the same in both places.

## List box colección o entity selection

In this type of list box, each column must be associated to an expression. The contents of each row is then evaluated per collection element or per entity of the entity selection.

Each element of the collection or each entity is available as an object that can be accessed through the [This](#) keyword. A column expression can be a property path, a project method, a variable, or any formula, accessing each entity or collection element object through [This](#), for example [This.<propertyPath>](#) (or [This.value](#) in case of a collection of scalar values). You can use the [LISTBOX SET COLUMN FORMULA](#) and [LISTBOX INSERT COLUMN FORMULA](#) commands to modify columns programmatically.

When the data source is an entity selection, any modifications made on the list box side are automatically saved in the

database. On the other hand, modifications made on the database side are visible in the list box after touched entities have been reloaded.

When the data source is a collection, any modifications made in the list box values are reflected in the collection. When the data source is a collection, any modifications made in the list box values are reflected in the collection. Por ejemplo:

```
myCol:=myCol.push("new value") //display new value in list box
```

## Propiedades soportadas

Las propiedades soportadas dependen del tipo de list box.

Propiedad	List box array	List box selección	List box colección o entity selection
Color de fondo alternado	X	X	X
Color de fondo	X	X	X
Negrita	X	X	X
Expresión color de fondo		X	X
Estilo del borde	X	X	X
Abajo	X	X	X
Class	X	X	X
Collection o entity selection		X	X
Redimensionamiento columnas auto	X	X	X
Elemento actual			X
Posición elemento actual			X
Fuente de datos	X	X	X
Nombre formulario detallado		X	
Mostrar encabezados	X	X	X
Mostrar pies	X	X	X
Doble clic en línea		X	
Arrastrable	X	X	X
Soltable	X	X	X
Focusable	X	X	X
Fuente	X	X	X
Color de fuente	X	X	X
Expresión color fuente		X	X
Tamaño fuente	X	X	X
Alto (list box)	X	X	X
Alto (encabezados)	X	X	X
Alto (pies)	X	X	X
Ocultar líneas vacías finales	X	X	X
Ocultar rectángulo de enfoque	X	X	X
Ocultar resultado selección	X	X	X
List box jerárquico	X		

Propiedad resaltado	List box array	List box selección	List box colección o entity selection
Alineación horizontal	X	X	X
Color líneas horizontales	X	X	X
Barra de desplazamiento horizontal	X	X	X
Dimensionamiento horizontal	X	X	X
Itálica	X	X	X
Izquierda	X	X	X
Tabla principal		X	
Meta info expression			X
Método	X	X	X
Líneas desplazables	X		
Selección temporal		X	
Número de columnas	X	X	X
Número de columnas bloqueadas	X	X	X
Número de columnas estáticas	X	X	X
Nombre del objeto	X	X	X
Derecha	X	X	X
Array colores de fondo	X		
Array de control de líneas	X		
Array colores de fuente	X		
Altura de las líneas	X		
Array altura de las líneas	X		
Array de estilos	X		
Elementos seleccionados			X
Modo de selección	X	X	X
Edición con un solo clic	X	X	X
Ordenable	X	X	X
Acción estándar	X		
Expresión estilo		X	X
Arriba	X	X	X
Transparente	X	X	X
Tipo	X	X	X
Subrayado	X	X	X
Variable o expresión	X	X	
Alineamiento vertical	X	X	X
Color líneas verticales	X	X	X
Barra de desplazamiento vertical	X	X	X
Dimensionamiento vertical	X	X	X
Visibilidad	X	X	X

Las columnas, los encabezados y los pies de list box soportan propiedades específicas.

## Eventos formulario soportados

Evento formulario	Propiedades adicionales devueltas (ver <a href="#">Evento formulario</a> para las propiedades principales)	Comentarios
On After Edit	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	
On After Keystroke	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	
On After Sort	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">headerName</a></li> </ul>	<i>Las fórmulas compuestas no se pueden ordenar. (e.g., This.firstName + This.lastName)</i>
On Alternative Click	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	<i>List box array únicamente</i>
On Before Data Entry	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	
On Before Keystroke	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	
On Begin Drag Over	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	
On Clicked	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	
On Close Detail	<ul style="list-style-type: none"> <li>• <a href="#">row</a></li> </ul>	<i>List box Selección actual &amp; Selección temporal únicamente</i>
On Collapse	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	<i>List box jerárquicos únicamente</i>
On Column Moved	<ul style="list-style-type: none"> <li>• <a href="#">columnName</a></li> <li>• <a href="#">newPosition</a></li> <li>• <a href="#">oldPosition</a></li> </ul>	
On Column Resize	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">newSize</a></li> <li>• <a href="#">oldSize</a></li> </ul>	

Eventos		
Evento formulario	Propiedades adicionales devueltas (ver <a href="#">Evento formulario</a> para las propiedades principales)	Comentarios
Change	<ul style="list-style-type: none"> <li>● <a href="#">columnName</a></li> <li>● <a href="#">row</a></li> </ul>	
On Delete Action	<ul style="list-style-type: none"> <li>● <a href="#">row</a></li> </ul>	
On Display Detail	<ul style="list-style-type: none"> <li>● <a href="#">isRowSelected</a></li> <li>● <a href="#">row</a></li> </ul>	
On Double Clicked	<ul style="list-style-type: none"> <li>● <a href="#">column</a></li> <li>● <a href="#">columnName</a></li> <li>● <a href="#">row</a></li> </ul>	
On Drag Over	<ul style="list-style-type: none"> <li>● <a href="#">area</a></li> <li>● <a href="#">areaName</a></li> <li>● <a href="#">column</a></li> <li>● <a href="#">columnName</a></li> <li>● <a href="#">row</a></li> </ul>	
On Drop	<ul style="list-style-type: none"> <li>● <a href="#">column</a></li> <li>● <a href="#">columnName</a></li> <li>● <a href="#">row</a></li> </ul>	
On Expand	<ul style="list-style-type: none"> <li>● <a href="#">column</a></li> <li>● <a href="#">columnName</a></li> <li>● <a href="#">row</a></li> </ul>	<i>List box jerárquicos únicamente</i>
On Footer Click	<ul style="list-style-type: none"> <li>● <a href="#">column</a></li> <li>● <a href="#">columnName</a></li> <li>● <a href="#">footerName</a></li> </ul>	<i>List box Arrays, Selección actual y Selección temporal únicamente</i>
On Getting Focus	<ul style="list-style-type: none"> <li>● <a href="#">column</a></li> <li>● <a href="#">columnName</a></li> <li>● <a href="#">row</a></li> </ul>	<i>Propiedades adicionales devueltas sólo al editar una celda</i>
On Header Click	<ul style="list-style-type: none"> <li>● <a href="#">column</a></li> <li>● <a href="#">columnName</a></li> <li>● <a href="#">headerName</a></li> </ul>	
On Load		
On Losing Focus	<ul style="list-style-type: none"> <li>● <a href="#">column</a></li> <li>● <a href="#">columnName</a></li> <li>● <a href="#">row</a></li> </ul>	<i>Propiedades adicionales devueltas sólo cuando la modificación de una celda se completa</i>
On Mouse Enter	<ul style="list-style-type: none"> <li>● <a href="#">area</a></li> <li>● <a href="#">areaName</a></li> <li>● <a href="#">column</a></li> <li>● <a href="#">columnName</a></li> <li>● <a href="#">row</a></li> </ul>	
On Mouse Leave		
On Mouse	<ul style="list-style-type: none"> <li>● <a href="#">area</a></li> </ul>	

Move Evento formulario	• <a href="#">areaName</a> Propiedades adicionales devueltas (ver <a href="#">Evento formulario</a> para las propiedades principales)	Comentarios
	• <a href="#">columnName</a> • <a href="#">row</a>	
On Open Detail	• <a href="#">row</a>	<i>List box Selección actual &amp; Selección temporal únicamente</i>
On Row Moved	• <a href="#">newPosition</a> • <a href="#">oldPosition</a>	<i>List box array únicamente</i>
On Selection Change		
On Scroll	• <a href="#">horizontalScroll</a> • <a href="#">verticalScroll</a>	
On Unload		

### Propiedades adicionales

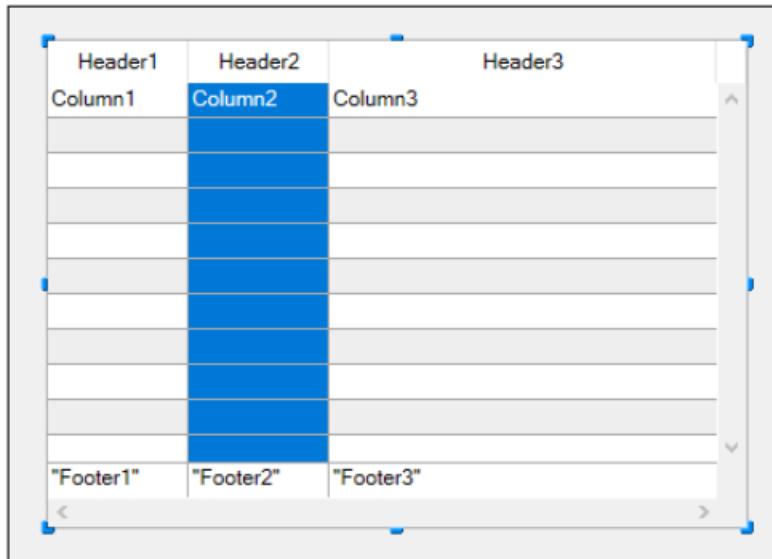
Los eventos formulario de los objetos list box o columnas de list box pueden devolver las siguientes propiedades adicionales:

Propiedad	Tipo	Descripción
area	texto	Área de objeto list box ("header", "footer", "cell")
areaName	texto	Nombre del área
column	entero largo	Número de columna
columnName	texto	Nombre de la columna
footerName	texto	Nombre del pie
headerName	texto	Nombre del encabezado
horizontalScroll	entero largo	Positive if scroll is towards the right, negative if towards the left
isRowSelected	booleano	True si la línea está seleccionada, de lo contrario False
newPosition	entero largo	Nueva posición de la columna o línea
newSize	entero largo	Nuevo tamaño (en píxeles) de la columna o línea
oldPosition	entero largo	Posición anterior de la columna o línea
oldSize	entero largo	Tamaño anterior (en píxeles) de la columna o línea
row	entero largo	Número de línea
verticalScroll	entero largo	Positive if scroll is towards the bottom, negative if towards the top

If an event occurs on a "fake" column or row that doesn't exist, an empty string is typically returned.

## Columnas de list box

A list box is made of one or more column object(s) which have specific properties. You can select a list box column in the Form editor by clicking on it when the list box object is selected:



You can set standard properties (text, background color, etc.) for each column of the list box; these properties take priority over those of the list box object properties.

You can define the [Expression type](#) for array list box columns (String, Text, Number, Date, Time, Picture, Boolean, or Object).

## Propiedades específicas de la columna

[Alpha Format](#) - [Alternate Background Color](#) - [Automatic Row Height](#) - [Background Color](#) - [Background Color Expression](#) - [Bold](#) - [Choice List](#) - [Class](#) - [Data Type \(selection and collection list box column\)](#) - [Date Format](#) - [Default Values](#) - [Display Type](#) - [Enterable](#) - [Entry Filter](#) - [Excluded List](#) - [Expression](#) - [Expression Type \(array list box column\)](#) - [Font](#) - [Font Color](#) - [Horizontal Alignment](#) - [Italic](#) - [Invisible](#) - [Maximum Width](#) - [Method](#) - [Minimum Width](#) - [Multi-style](#) - [Number Format](#) - [Object Name](#) - [Picture Format](#) - [Resizable](#) - [Required List](#) - [Row Background Color Array](#) - [Row Font Color Array](#) - [Row Style Array](#) - [Save as](#) - [Style Expression](#) - [Text when False/Text when True](#) - [Time Format](#) - [Truncate with ellipsis](#) - [Underline](#) - [Variable or Expression](#) - [Vertical Alignment](#) - [Width](#) - [Wordwrap](#)

## Eventos formulario soportados

Evento formulario	Propiedades adicionales devueltas (ver <a href="#">Evento formulario</a> para las propiedades principales)	Comentarios
On After Edit	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	
On After Keystroke	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	
On After Sort	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">headerName</a></li> </ul>	<i>Las fórmulas compuestas no se pueden ordenar. (e.g., This.firstName + This.lastName)</i>
On Alternative Click	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	<i>List box array únicamente</i>
On Before Data Entry	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	

Evento Keystroke	<ul style="list-style-type: none"> <li>• <a href="#">Properties adicionales devueltas (ver Evento formulario para las propiedades principales)</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	Comentarios
On Begin Drag Over	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	
On Clicked	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	
On Column Moved	<ul style="list-style-type: none"> <li>• <a href="#">columnName</a></li> <li>• <a href="#">newPosition</a></li> <li>• <a href="#">oldPosition</a></li> </ul>	
On Column Resize	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">newSize</a></li> <li>• <a href="#">oldSize</a></li> </ul>	
On Data Change	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	
On Double Clicked	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	
On Drag Over	<ul style="list-style-type: none"> <li>• <a href="#">area</a></li> <li>• <a href="#">areaName</a></li> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	
On Drop	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	
On Footer Click	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">footerName</a></li> </ul>	<i>List box Arrays, Selección actual y Selección temporal únicamente</i>
On Getting Focus	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	<i>Propiedades adicionales devueltas sólo al editar una celda</i>
On Header Click	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">headerName</a></li> </ul>	
On Load		
On Losing Focus	<ul style="list-style-type: none"> <li>• <a href="#">column</a></li> <li>• <a href="#">columnName</a></li> <li>• <a href="#">row</a></li> </ul>	<i>Propiedades adicionales devueltas sólo cuando la modificación de una celda se completa</i>

Evento	<ul style="list-style-type: none"><li>● <a href="#">row</a></li></ul> Propiedades adicionales devueltas (ver <a href="#">Evento formulario</a> para las propiedades principales)	Comentarios
On Row Moved	<ul style="list-style-type: none"><li>● <a href="#">newPosition</a></li></ul>	<i>List box array únicamente</i>
On Scroll	<ul style="list-style-type: none"><li>● <a href="#">horizontalScroll</a></li><li>● <a href="#">verticalScroll</a></li></ul>	
On Unload		

## Encabezados de list box

Para poder acceder a las propiedades de los pies de un list box, debe activar la opción **Mostrar pies**.

Cuando se muestran los encabezados, puede seleccionar un encabezado en el editor de formularios haciendo clic en él cuando el objeto List box esté seleccionado:

Puede definir propiedades de texto estándar para cada encabezado de columna de List box; en este caso, estas propiedades tienen prioridad sobre las de la columna o del propio List box.

Además, tiene acceso a las propiedades específicas de los encabezados. Specifically, an icon can be displayed in the header next to or in place of the column title, for example when performing [customized sorts](#).

Name	Position
Smith	Administrator

At runtime, events that occur in a header are generated in the [list box column object method](#).

When the `OBJECT SET VISIBLE` command is used with a header, it is applied to all headers, regardless of the individual element set by the command. For example, `OBJECT SET VISIBLE(*;"header3";False)` will hide all headers in the list box object to which *header3* belongs and not simply this header.

## Propiedades específicas de los encabezados

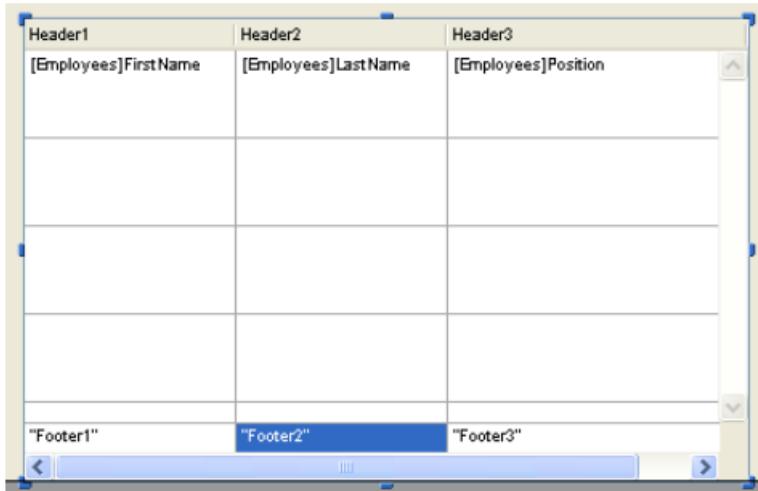
Negrita - Clase Css - Fuente - Color de fuente - Mensaje de ayuda - Alineación horizontal - Icon Location - Itálica - Nombre de objeto - Ruta de acceso - Título - Subrayado - Variable o Expresión - Alineación vertical - Ancho

## Pies de list box

Para poder acceder a las propiedades de los encabezados de un list box, debe activar la opción [Mostrar encabezados](#) del list box.

Los List box pueden contener "pies de página" no editables, que muestren información adicional. En el caso de los datos mostrados en forma de tabla, los pies de página suelen utilizarse para mostrar cálculos como los totales o los promedios.

Cuando se muestran los pies, puede hacer clic para seleccionar un pie de list box en el editor de formularios haciendo clic en el objeto:



Para cada pie de columna de list box, puede definir propiedades de texto estándar: en este caso, estas propiedades tienen prioridad sobre las de la columna o del list box. You can also access specific properties for footers. In particular, you can insert a [custom or automatic calculation](#).

At runtime, events that occur in a footer are generated in the [list box column object method](#).

When the `OBJECT SET VISIBLE` command is used with a footer, it is applied to all footers, regardless of the individual element set by the command. For example, `OBJECT SET VISIBLE(*;"footer3";False)` will hide all footers in the list box object to which *footer3* belongs and not simply this footer.

## Propiedades específicas de los pies

[Alpha Format](#) - [Background Color](#) - [Bold](#) - [Class](#) - [Date Format](#) - [Expression Type](#) - [Font](#) - [Font Color](#) - [Help Tip](#) - [Horizontal Alignment](#) - [Italic](#) - [Number Format](#) - [Object Name](#) - [Picture Format](#) - [Time Format](#) - [Truncate with ellipsis](#) - [Underline](#) - [Variable Calculation](#) - [Variable or Expression](#) - [Vertical Alignment](#) - [Width](#) - [Wordwrap](#)

## Gestión de entrada

For a list box cell to be enterable, both of the following conditions must be met:

- The cell's column must have been set as [Enterable](#) (otherwise, the cells of the column can never be enterable).
- En el evento [On Before Data Entry](#), \$0 no devuelve -1. When the cursor arrives in the cell, the [On Before Data Entry](#) event is generated in the column method. Si, en el contexto de este evento, \$0 se define como -1, la celda se considera como no editable. Si el evento se generó después de presionar Tab o Mayús+Tab, el foco pasa a la siguiente celda o a la anterior, respectivamente. Si \$0 no es -1 (por defecto \$0 es 0), la celda se puede introducir y pasa al modo de edición.

Let's consider the example of a list box containing two arrays, one date and one text. The date array is not enterable but the text array is enterable if the date has not already past.

Header1	Header2
Variable Name: tDate	Variable Name: tText

Here is the method of the *arrText* column:

```

Case of
  :(FORM event.code=On Before Data Entry) // una celda obtiene el foco
    LISTBOX GET CELL POSITION(*;"lb";$col;$row)
  // identification of cell
    If(arrDate{$row}<Current date) // si la fecha es anterior a hoy
      $0:=-1 // la celda NO es editable
    Else
      // de lo contrario, la celda es editable
    End if
End case

```

The `On Before Data Entry` event is returned before `On Getting Focus`.

In order to preserve data consistency for selection type and entity selection type list boxes, any modified record/entity is automatically saved as soon as the cell is validated, i.e.:

- when the the cell is deactivated (user presses tab, clicks, etc.)
- cuando el listbox ya no tiene el foco,
- cuando el formulario ya no tiene el foco.

The typical sequence of events generated during data entry or modification is as follows:

Acción	Tipo(s) de Listbox	Secuencia de eventos
A cell switches to edit mode (user action or a call to the <code>EDIT ITEM</code> command)	Todos	On Before Data Entry
	Todos	On Getting Focus
Cuando se ha editado el valor de una celda	Todos	On Before Keystroke
	Todos	On After Keystroke
	Todos	On After Edit
Un usuario valida y abandona la celda	List box de tipo selección	Guardar
	List box de tipo selección de registro	Activación de On saving an existing record (si definido)
	List box de tipo selección	On Data Change(*)
	List box de tipo selección de entidades	Entity is saved with automerge option, optimistic lock (see <code>entity.save()</code> ). In case of successful save, the entity is refreshed with the last update done. Si la operación de guardado falla, se mostrará un error
	Todos	On Losing Focus

(\*) With entity selection list boxes, in the `On Data Change` event:

- the `Current item` object contains the value before modification.
- the `This` object contains the modified value.

Data entry in collection/entity selection type list boxes has a limitation when the expression evaluates to null. In this case, it is not possible to edit or remove the null value in the cell.

## Gestión de selecciones

La gestión de selecciones es diferente dependiendo de si el list box se basa en un array, en una selección de registros o en una selección de colecciones/entidades:

- **Lista box de tipo selección:** las selecciones se gestionan mediante un conjunto llamado por defecto `$ListboxSetX` (donde X empieza en 0 y se incrementa en función del número de list box en el formulario), que puede modificar si es necesario. Este conjunto se [define en las propiedades](#) del list box. Es mantenido automáticamente por 4D: si el usuario selecciona una o más líneas en el list box, el conjunto se actualiza inmediatamente. Por otra parte, también es posible utilizar los comandos del tema "Conjuntos" para modificar por programación la selección en el list box.
- **List box de tipo colección/selección de entidades :** las selecciones se gestionan a través de las propiedades del list box dedicado:
  - [Elemento actual](#) es un objeto que recibirá el elemento/la entidad seleccionado(a)
  - [Elementos seleccionados](#) es una colección de elementos seleccionados
  - [Posición del elemento actual](#) devuelve la posición del elemento o de la entidad seleccionada.
- **List box de tipo array :** el comando `LISTBOX SELECT ROW` puede utilizarse para seleccionar una o más líneas del list box por programación. La [variable asociada al objeto List box](#) se utiliza para obtener, definir o almacenar las selecciones de líneas en el objeto. Esta variable corresponde a un array de booleanos que es creado y mantenido automáticamente por 4D. El tamaño de este array viene determinado por el tamaño del list box: contiene el mismo número de elementos que el array más pequeño asociado a las columnas. Cada elemento de este array contiene `True` si se selecciona la línea correspondiente y `False` en caso contrario. 4D actualiza el contenido de este array en función de las acciones del usuario. Por el contrario, puede cambiar el valor de los elementos del array para cambiar la selección en el list box. Por otra parte, no se pueden insertar ni borrar líneas en este array; tampoco se pueden reescribir las líneas. El comando `Count in array` puede utilizarse para averiguar el número de líneas seleccionadas. Por ejemplo, este método permite invertir la selección de la primera línea del list box (tipo array):

```
ARRAY BOOLEAN(tListBox;10)
//tListBox es el nombre de la variable asociada al list box en el formulario
If(tListBox{1}=True)
  tListBox{1}:=False
Else
  tListBox{1}:=True
End if
```

The `OBJECT SET SCROLL POSITION` command scrolls the list box rows so that the first selected row or a specified row is displayed.

## Personalizar la apariencia de las líneas seleccionadas

When the [Hide selection highlight](#) option is selected, you need to make list box selections visible using available interface options. Since selections are still fully managed by 4D, this means:

- For array type list boxes, you must parse the Boolean array variable associated with the list box to determine which rows are selected or not.
- For selection type list boxes, you have to check whether the current record (row) belongs to the set specified in the [Highlight Set](#) property of the list box.

You can then define specific background colors, font colors and/or font styles by programming to customize the appearance of selected rows. This can be done using arrays or expressions, depending on the type of list box being displayed (see the following sections).

You can use the `lk_inherited` constant to mimic the current appearance of the list box (e.g., font color, background color, font style, etc.).

## List box de tipo selección

To determine which rows are selected, you have to check whether they are included in the set indicated in the [Highlight Set](#) property of the list box. You can then define the appearance of selected rows using one or more of the relevant [color or style expression property](#).

Keep in mind that expressions are automatically re-evaluated each time the:

- la selección de list box cambia.
- list box obtiene o pierde el foco.
- form window containing the list box becomes, or ceases to be, the frontmost window.

## List box de tipo array

You have to parse the Boolean array [Variable or Expression](#) associated with the list box to determine whether rows are selected or not selected.

You can then define the appearance of selected rows using one or more of the relevant [color or style array property](#).

Note that list box arrays used for defining the appearance of selected rows must be recalculated during the [On Selection Change](#) form event; however, you can also modify these arrays based on the following additional form events:

- [On Getting Focus](#) (propiedad list box)
- [On Losing Focus](#) (propiedad list box)
- [On Activate](#) (propiedad list box)
- [On Deactivate](#) (form property) ...depending on whether and how you want to visually represent changes of focus in selections.

## Ejemplo

You have chosen to hide the system highlight and want to display list box selections with a green background color, as shown here:

Category	ID	Reference	Value
Alpha	215	5A0DF64-EC5-955F7EA-BD284E4-8A	15,425
Bravo	196	D9E3484-547-AEECB8B-B1808FF-A6	4,592
Alpha	205	3295824-3A8-B48B870-2074C57-B9	-3,672
Charlie	197	B3800C4-C64-A6C95CB-ED27729-B5	16,212
Echo	214	835F344-8EE-B422E66-5C52074-01	-12,332
Alpha	200	46784B4-B20-AE2E51D-0159EA4-D0	1,283
Delta	213	11F5FD4-E48-9BD6E93-E8B9F82-59	13,236
Delta	203	3E80494-879-9F2CEC2-4008AA4-F5	-12,231
Charlie	202	015D694-9C8-91113AA-B8A51A1-52	27,100
Bravo	211	E998AC4-FAE-93BE025-E4CA634-E8	2,630
Charlie	207	B19F244-A30-A03B668-C407B43-D4	16,677
Delta	208	41B1DE4-D29-BC8E7BF-5062D92-B7	-14,759
Echo	199	7005654-722-926CDCE-D8E1BBD-83	23,952
Delta	198	0AD0734-0C7-BA8168E-A0AB67A-1A	-19,758
Alpha	210	6F46794-0D6-AF0E61A-D43231E-3E	24,342
Bravo	201	00A8334-B4B-8419285-8772CFB-B4	-3,657
Charlie	212	9EF2FD4-B1B-97138DE-B3750BA-FB	-4,850
Echo	209	FD05424-365-B8DB0C2-91098A8-80	2,941
Echo	204	7473724-2FA-82F49A5-010BDED-98	22,200
Bravo	206	3537D34-A9A-AE41C4E-34EC5B6-43	1,205

For an array type list box, you need to update the [Row Background Color Array](#) by programming. In the JSON form, you have defined the following Row Background Color Array for the list box:

```
"rowFillSource": "_ListboxBackground",
```

In the object method of the list box, you can write:

```
Case of
:(FORM event.code=On Selection Change)
$n:=Size of array(LB_Arrays)
ARRAY LONGINT(_ListboxBackground;$n) // colores de fondo de la línea
For($i;1;$n)
  If(LB_Arrays{$i}=True) // selected
    _ListboxBackground{$i}:=0x0080C080 // fondo verde
  Else // not selected
    _ListboxBackground{$i}:=lk inherited
  End if
End for
End case
```

For a selection type list box, to produce the same effect you can use a method to update the [Background Color Expression](#) based on the set specified in the [Highlight Set](#) property.

For example, in the JSON form, you have defined the following Highlight Set and Background Color Expression for the list box:

```
"highlightSet": "$SampleSet",
"rowFillSource": "UI_SetColor",
```

You can write in the *UI\_SetColor* method:

```
If(Is in set("$SampleSet"))
  $color:=0x0080C080 // green background
Else
  $color:=lk inherited
End if

$0:=$color
```

In hierarchical list boxes, break rows cannot be highlighted when the [Hide selection highlight](#) option is checked. Since it is not possible to have distinct colors for headers of the same level, there is no way to highlight a specific break row by programming.

## Gestión de ordenaciones

Un orden en un list box puede ser estándar o personalizado. When a column of a list box is sorted, all other columns are always synchronized automatically.

### Ordenación estándar

By default, a list box provides standard column sorts when the header is clicked. A standard sort is an alphanumeric sort of evaluated column values, alternately ascending/descending with each successive click.

You can enable or disable standard user sorts by disabling the [Sortable](#) property of the list box (enabled by default).

Standard sort support depends on the list box type:

Tipo de list box	Support of standard sort	Comentarios
Colección de objetos	Sí	<ul style="list-style-type: none"> <li>Las columnas "This.a" o "This.a.b" son ordenables.</li> <li>The <a href="#">list box source property</a> must be an <a href="#">assignable expression</a>.</li> </ul>
Colección de valores escalares	No	Use custom sort with <a href="#">orderBy()</a> function
Entity selection	Sí	<ul style="list-style-type: none"> <li>The <a href="#">list box source property</a> must be an <a href="#">assignable expression</a>.</li> <li>Supported: sorts on object attribute properties (e.g. "This.data.city" when "data" is an object attribute)</li> <li>Supported: sorts on related attributes (e.g. "This.company.name")</li> <li>Not supported: sorts on object attribute properties through related attributes (e.g. "This.company.data.city"). For this, you need to use custom sort with <a href="#">orderByFormula()</a> function (see example below)</li> </ul>
Selección actual	Sí	Only simple expressions are sortable (e.g. [Table_1]Field_2 )
Named selection	No	
Arrays	Sí	Columns bound to picture and pointer arrays are not sortable

## Ordenación personalizada

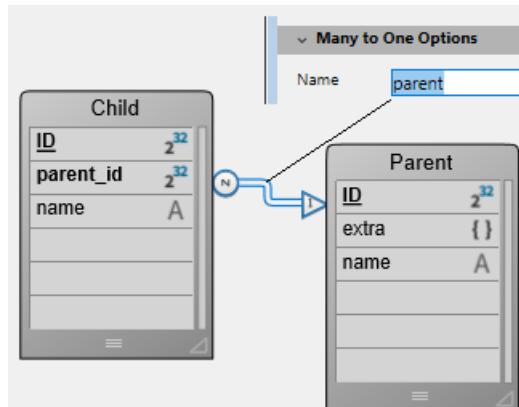
The developer can set up custom sorts, for example using the [LISTBOX SORT COLUMNS](#) command and/or combining the [On Header Click](#) and [On After Sort](#) form events and relevant 4D commands.

Los ordenamientos personalizados le permiten:

- carry out multi-level sorts on several columns, thanks to the [LISTBOX SORT COLUMNS](#) command,
- use functions such as [collection.orderByFormula\(\)](#) or [entitySelection.orderByFormula\(\)](#) to sort columns on complex criteria.

### Ejemplo

You want to sort a list box using values of a property stored in a related object attribute. Tiene la siguiente estructura:



You design a list box of the entity selection type, bound to the `Form.child` expression. In the `On Load` form event, you execute `Form.child:=ds.Child.all()`.

You display two columns:

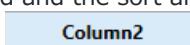
Nombre del hijo	Apodo del padre
This.name	This.parent.extra.nickname

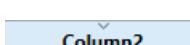
If you want to sort the list box using the values of the second column, you have to write:

```
If (Form event code=On Header Click)
    Form.child:=Form.child.orderByFormula("This.parent.extra.nickname"; dk ascending)
End if
```

## Variable de encabezado de columna

The value of the [column header variable](#) allows you to manage additional information: the current sort of the column (read) and the display of the sort arrow.

- If the variable is set to 0, the column is not sorted and the sort arrow is not displayed.  

- Si la variable está definida como 1, la columna se organiza en orden ascendente y se muestra la flecha de ordenación.  

- Si la variable está definida en 2, la columna se organiza en orden descendente y se muestra la flecha de clasificación.  


Only declared or dynamic [variables](#) can be used as header column variables. Other kinds of [expressions](#) such as `Form.sortValue` are not supported.

You can set the value of the variable (for example, `Header2:=2`) in order to "force" the sort arrow display. The column sort itself is not modified in this case; it is up to the developer to handle it.

The [OBJECT SET FORMAT](#) command offers specific support for icons in list box headers, which can be useful when you want to work with a customized sort icon.

## Gestión de los colores, estilos y visualización de las líneas

There are several different ways to set background colors, font colors and font styles for list boxes:

- at the level of the [list box object properties](#),
- at the level of the [column properties](#),
- using [arrays or expressions properties](#) for the list box and/or for each column,
- at the level of the text of each cell (if [multi-style text](#)).

## Prioridad & herencia

Priority and inheritance principles are observed when the same property is set at more than one level.

Nivel de prioridad	Ubicación del parámetro
alta prioridad	Celda (si texto multiestilo)
	Arrays de columnas/métodos
	Arrays/métodos de Listbox
	Propiedades de la columna
	Propiedades de list box
baja prioridad	Meta Info expression (for collection or entity selection list boxes)

For example, if you set a font style in the list box properties and another using a style array for the column, the latter one will be taken into account.

For each attribute (style, color and background color), an inheritance is implemented when the default value is used:

- para los atributos de las celdas: valores de atributos de las líneas
- para los atributos líneas: valores de atributos de columnas
- for column attributes: attribute values of the list box

This way, if you want an object to inherit the attribute value from a higher level, you can use pass the `lk_inherited` constant (default value) to the definition command or directly in the element of the corresponding style/color array. For example, given an array list box containing a standard font style with alternating colors:

standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard

Realiza las siguientes modificaciones:

- change the background of row 2 to red using the [Row Background Color Array](#) property of the list box object,
- change the style of row 4 to italics using the [Row Style Array](#) property of the list box object,
- two elements in column 5 are changed to bold using the [Row Style Array](#) property of the column 5 object,
- the 2 elements for column 1 and 2 are changed to dark blue using the [Row Background Color Array](#) property for the column 1 and 2 objects:

standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard

To restore the original appearance of the list box, you can:

- pass the `lk_inherited` constant in element 2 of the background color arrays for columns 1 and 2: then they inherit the red background color of the row.
- pass the `lk_inherited` constant in elements 3 and 4 of the style array for column 5: then they inherit the standard style, except for element 4, which changes to italics as specified in the style array of the list box.
- pass the `lk_inherited` constant in element 4 of the style array for the list box in order to remove the italics style.
- pass the `lk_inherited` constant in element 2 of the background color array for the list box in order to restore the original alternating color of the list box.

## Uso de arrays y expresiones

Depending of the list box type, you can use different properties to customize row colors, styles and display:

Propiedad	List box array	List box selección	List box colección o entity selection
Color de fondo	Array colores de fondo	Expresión color de fondo	Background Color Expression or Meta info expression
Color de fuente	Array colores de fuente	Expresión color fuente	Font Color Expression or Meta info expression

[Row Style Array](#) | [Style Expression](#) | [Style Expression or Meta info expression](#) | [Display](#) | [Row Control Array](#) | -|-

## Imprimir list boxes

Two printing modes are available: preview mode - which can be used to print a list box like a form object, and advanced mode - which lets you control the printing of the list box object itself within the form. Note that the "Printing" appearance is available for list box objects in the Form editor.

### Modo de vista previa

Printing a list box in preview mode consists of directly printing the list box and the form that contains it using the standard print commands or the Print menu command. El list box se imprime tal como está en el formulario. This mode does not allow precise control of the printing of the object; in particular, it does not allow you to print all the rows of a list box that contains more rows than it can display.

### Modo avanzado

In this mode, the printing of list boxes is carried out by programming, via the `Print object` command (project forms and table forms are supported). The `LISTBOX GET PRINT INFORMATION` command is used to control the printing of the object.

En este modo:

- The height of the list box object is automatically reduced when the number of rows to be printed is less than the original height of the object (there are no "blank" rows printed). On the other hand, the height does not automatically increase according to the contents of the object. The size of the object actually printed can be obtained via the `LISTBOX GET PRINT INFORMATION` command.
- The list box object is printed "as is", in other words, taking its current display parameters into account: visibility of headers and gridlines, hidden and displayed rows, etc. These parameters also include the first row to be printed: if you call the `OBJECT SET SCROLL POSITION` command before launching the printing, the first row printed in the list box will be the one designated by the command.
- An automatic mechanism facilitates the printing of list boxes that contain more rows than it is possible to display: successive calls to `Print object` can be used to print a new set of rows each time. The `LISTBOX GET PRINT INFORMATION` command can be used to check the status of the printing while it is underway.

## List box jerárquicos

Un list box jerárquico es un list box en el que el contenido de la primera columna aparece en forma jerárquica. Este tipo de representación se adapta a la presentación de información que incluye valores repetidos y/o que dependen jerárquicamente (país/región/ciudad, etc.).

Sólo los [list box de tipo array](#) pueden ser jerárquicos.

Los list box jerárquicos son una forma particular de representar los datos, pero no modifican la estructura de datos (arrays). Los list box jerárquicos se gestionan exactamente igual que los list box clásicos.

### Definir una jerarquía

Para definir un list box jerárquico, existen varias posibilidades:

- Configurar manualmente los elementos jerárquicos utilizando la lista de propiedades del editor de formularios (o

editar el formulario JSON).

- Generar visualmente la jerarquía utilizando el menú emergente de gestión de list box, en el editor de formularios.
- Use the [LISTBOX SET HIERARCHY](#) and [LISTBOX GET HIERARCHY](#) commands, described in the *4D Language Reference* manual.

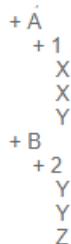
## Propiedades del List Box jerárquico

This property specifies that the list box must be displayed in hierarchical form. In the JSON form, this feature is triggered [when the \*dataSource\* property value is an array](#), i.e. a collection.

Additional options (Variable 1...10) are available when the *Hierarchical List Box* option is selected, corresponding to each *dataSource* array to use as break column. Each time a value is entered in a field, a new row is added. Se pueden especificar hasta 10 variables. These variables set the hierarchical levels to be displayed in the first column.

The first variable always corresponds to the name of the variable for the first column of the list box (the two values are automatically bound). This first variable is always visible and enterable. Por ejemplo: country. The second variable is also always visible and enterable; it specifies the second hierarchical level. Por ejemplo: regions. Beginning with the third field, each variable depends on the one preceding it. Por ejemplo: counties, cities, etc. A maximum of ten hierarchical levels can be specified. If you remove a value, the whole hierarchy moves up a level.

The last variable is never hierarchical even if several identical values exists at this level. For example, referring to the configuration illustrated above, imagine that arr1 contains the values A A A B B B, arr2 has the values 1 1 1 2 2 2 and arr3 the values X X Y Y Y Z. In this case, A, B, 1 and 2 could appear in collapsed form, but not X and Y:

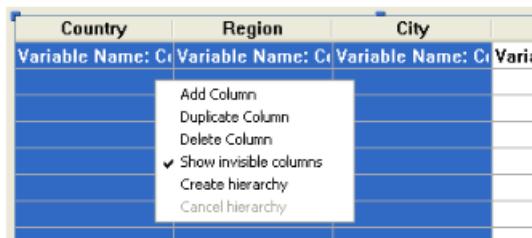


This principle is not applied when only one variable is specified in the hierarchy: in this case, identical values may be grouped.

If you specify a hierarchy based on the first columns of an existing list box, you must then remove or hide these columns (except for the first), otherwise they will appear in duplicate in the list box. If you specify the hierarchy via the pop-up menu of the editor (see below), the unnecessary columns are automatically removed from the list box.

## Crear una jerarquía mediante el menú contextual

When you select at least one column in addition to the first one in a list box object (of the array type) in the form editor, the Create hierarchy command is available in the context menu:



Este comando es un acceso directo para definir una jerarquía. Cuando se selecciona, se llevan a cabo las siguientes acciones:

- The Hierarchical list box option is checked for the object in the Property List.
- Las variables de las columnas se utilizan para definir la jerarquía. Reemplazan las variables ya definidas.
- Las columnas seleccionadas ya no aparecen en el list box (excepto el título de la primera).

Ejemplo: dado un list box cuyas primeras columnas contienen País, Región, Ciudad y Población. When Country, Region and City are selected, if you choose Create hierarchy in the context menu, a three-level hierarchy is created in the first column, columns 2 and 3 are removed and the Population column becomes the second:

The screenshot shows a software interface with two main components: a configuration dialog on the right and a preview window on the left.

**Configuration Dialog (PropertyList):**

- Type:** List Box
- Object Name:** List Box
- Variable or Expression:** List Box
- Data Source:** Arrays

**List Box Properties:**

- Number of Columns: 2
- Number of Locked Columns: 0
- Number of Static Columns: 1
- Row Control Array:
- Selection Mode: Multiple

**Headers:**

- Display Headers:
- Height: 1 lines

**Footers:**

- Display Footers:
- Height: 1 lines

**Hierarchy:**

- Hierarchical List Box:
- Variable 1: Country
- Variable 2: Region
- Variable 3: City (Selected)
- Variable 4:

**Other Sections:**

- Coordinates & Sizing
- Resizing Options
- Entry
- Display
- Appearance
- Background and Border
- Text

**Preview Window:**

A preview window titled "Country Population" displays a table with two columns: "Country" and "Population". The "Country" column has a header "Variable Name: Column1" and the "Population" column has a header "Variable Name: Column4". Both columns contain several rows of data.

## Cancelar jerarquía

When the first column is selected and already specified as hierarchical, you can use the Cancel hierarchy command. Cuando elige este comando, se llevan a cabo las siguientes acciones:

- The Hierarchical list box option is deselected for the object,
  - Los niveles jerárquicos 2 a X se eliminan y se transforman en columnas añadidas al list box.

## Principios de funcionamiento

Cuando se abre por primera vez un formulario que contiene un list box jerárquico, por defecto se despliegan todas las líneas.

Cuando los valores se repiten en los arrays, se añade automáticamente una línea de ruptura y un "nodo" jerárquico en el list box. Por ejemplo, imagine un list box que contenga cuatro arrays que indiquen las ciudades, cada una de ellas caracterizada por su país, su región, su nombre y su número de habitantes:

Country	Region	City	Population
France	Brittany	Rennes	200000
France	Brittany	Quimper	80000
France	Brittany	Brest	120000
France	Normandy	Caen	75000
France	Normandy	Deauville	35000

Si este list box se muestra en forma jerárquica (los tres primeros arrays están incluidos en la jerarquía), se obtiene:

City	Population
<b>France</b>	
<b>  Brittany</b>	
Rennes	200000
Quimper	80000
Brest	120000
<b>  Normandy</b>	
Caen	75000
Deauville	35000

Los arrays no se ordenan antes de construir la jerarquía. Si, por ejemplo, un array contiene los datos AAABBAACC, la jerarquía obtenida será:

```
> &gt; En este contexto, sólo la sintaxis que utiliza la variable array puede funcionar con los comandos
```

Para desplegar o contraer un "nodo" jerárquico, basta con hacer clic en él. If you Alt+click (Windows) or Option+click (macOS) on the node, all its sub-elements will be expanded or collapsed as well. Estas operaciones también pueden realizarse por programación utilizando los comandos `LISTBOX EXPAND` y `LISTBOX COLLAPSE`.

Cuando se incluyen valores del tipo fecha u hora en un list box jerárquico, se muestran en el formato del sistema corto.

### Ordenación en list box jerárquicos

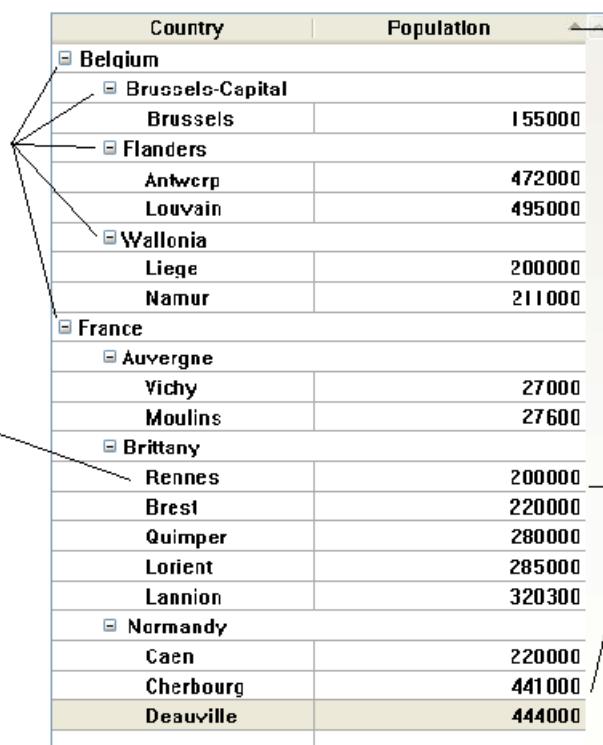
Cuando se incluyen valores del tipo fecha u hora en un list box jerárquico, se muestran en el formato del sistema corto.

- En primer lugar, todos los niveles de la columna jerárquica (primera columna) se clasifican automáticamente por orden ascendente.
- La ordenación se realiza por orden ascendente o descendente (según la acción del usuario) sobre los valores de la columna en la que se ha hecho clic.
- Todas las columnas son sincronizadas.
- En las siguientes ordenaciones realizadas en columnas no jerárquicas del list box, sólo se ordena el último nivel de la primera columna. Es posible modificar la ordenación de esta columna haciendo clic en su encabezado.

Cuando se incluyen valores del tipo fecha u hora en un list box jerárquico, se muestran en el formato del sistema corto.

Country	Population
<b>France</b>	
<b>  Brittany</b>	
Rennes	200000
Quimper	80000
Brest	120000
Lannion	20300
Lorient	35000
<b>  Normandy</b>	
Caen	220000
Deauville	4000
Cherbourg	41000
<b>  Auvergne</b>	
Vichy	27000
Moulins	20600
<b>  Belgium</b>	
<b>    Wallonia</b>	
Namur	111000
Liege	200000
<b>    Flanders</b>	
Antwerp	472000
Louvain	95000
<b>    Brussels-Capital</b>	
Brussels	155000

Si hace clic en el encabezado "Population" para ordenar las poblaciones por orden ascendente (o alternativamente descendente), los datos aparecen de la siguiente manera:



Country	Population
Belgium	
Brussels-Capital	
Brussels	155000
Flanders	
Antwerp	472000
Louvain	495000
Wallonia	
Liege	200000
Namur	211000
France	
Auvergne	
Vichy	27000
Moulins	27600
Brittany	
Rennes	200000
Brest	220000
Quimper	280000
Lorient	285000
Lannion	320300
Normandy	
Caen	220000
Cherbourg	441000
Deauville	444000

Como para todos los list box, puede [desactivar el mecanismo de ordenación estándar](#) y gestionar las ordenaciones por programación.

### Selecciones y posiciones en list box jerárquicos

Un list box jerárquico muestra un número variable de líneas en la pantalla según el estado desplegado/contraído de los nodos jerárquicos. Sin embargo, esto no significa que el número de líneas de los arrays varíe. Sólo se modifica la visualización, no los datos. Es importante entender este principio porque la gestión programada de los list box jerárquicos se basa siempre en los datos de los arrays, no en los datos mostrados. En particular, las filas de ruptura añadidas automáticamente no se tienen en cuenta en los arrays de opciones de visualización (ver más adelante).

Veamos, por ejemplo, los siguientes arrays:

France	Brittany	Brest
France	Brittany	Quimper
France	Brittany	Rennes

Si estos arrays se representan jerárquicamente, la línea "Quimper" no se mostrará en la segunda línea, sino en la cuarta, debido a las dos líneas de ruptura que se añaden:

France
Brittany
Brest
Quimper
Rennes

Independientemente de cómo se muestren los datos en el list box (de forma jerárquica o no), si quiere cambiar la línea que contiene "Quimper" a negrita, debe utilizar la instrucción `Style{2} = bold`. Sólo se tiene en cuenta la posición de la línea en los arrays.

Este principio se aplica a los arrays internos que se pueden utilizar para gestionar:

- colores
- colores de fondo
- estilos
- líneas ocultas
- selecciones

Este principio se aplica a los arrays internos que se pueden utilizar para gestionar:

```
->MyListbox{3}:=True
```

Este principio se aplica a los arrays internos que se pueden utilizar para gestionar:

France
Brittany
Brest
Quimper
Rennes

Si una o más líneas están ocultas porque sus padres están contraídos, ya no se seleccionan. Sólo se pueden seleccionar las líneas visibles (directamente o por desplazamiento). En otras palabras, las líneas no pueden estar ocultas y seleccionadas a la vez.

Al igual que con las selecciones, el comando `LISTBOX GET CELL POSITION` devolverá los mismos valores para un list box jerárquico y un list box no jerárquico. Esto significa que en los dos ejemplos siguientes, `LISTBOX GET CELL POSITION` devolverá la misma posición: (3;2).

*Representación jerárquica:*

France	Brittany	Brest	120000
France	Brittany	Quimper	80000
France	Brittany	Rennes	200000
France	Normandy	Caen	75000

*Representación no jerárquica:*

France	
Brittany	
Brest	120000
Quimper	80000
Rennes	200000
Normandy	
Caen	75000

When all the rows of a sub-hierarchy are hidden, the break line is automatically hidden. En el ejemplo anterior, si las líneas 1 a 3 están ocultas, la línea de ruptura "Bretaña" no aparecerá.

Líneas de quiebre

Si el usuario selecciona una línea de ruptura, `LISTBOX GET CELL POSITION` devuelve la primera ocurrencia de la línea en el array correspondiente. En el caso siguiente:

France	
Brittany	
Brest	120000
Quimper	80000
Rennes	200000
Normandy	
Caen	75000

... `LISTBOX GET CELL POSITION` devuelve (2;4). Para seleccionar una línea de ruptura por programación, deberá utilizar el comando `LISTBOX SELECT BREAK`.

Las líneas de rotura no se tienen en cuenta en los arrays internos utilizados para gestionar el aspecto gráfico de los list box (estilos y colores). No obstante, es posible modificar estas características para las líneas de ruptura mediante los comandos de gestión gráfica de los objetos. Basta con ejecutar los comandos adecuados en los arrays que constituyen la jerarquía.

El siguiente list box fue diseñado utilizando un array de objetos:

*Representación jerárquica:*

(T1)	(T2)	(T3)	(T4)	(tStyle)	(tColor)
France	Brittany	Brest	120000	Normal	0
France	Brittany	Quimper	80000	Underline	0
France	Brittany	Rennes	200000	Normal	0xFF0000
France	Normandy	Caen	220000	Normal	0
France	Normandy	Deauville	4000	Normal	0

Representación no jerárquica:

France		
Brittany		
Brest	120000	
Quimper	80000	
Rennes	200000	
Normandy		
Caen	75000	
Deauville	4000	

En modo jerárquico, los niveles de ruptura no son tenidos en cuenta por los arrays de modificación de estilo denominados `tStyle` y `tColors`. Para modificar el color o el estilo de los niveles de ruptura, debe ejecutar las siguientes instrucciones:

```
OBJECT SET RGB COLORS(T1;0x0000FF;0xB0B0B0)
OBJECT SET FONT STYLE(T2;Bold)
```

> En este contexto, sólo la sintaxis que utiliza la variable array puede funcionar con los comandos de la propiedad del objeto porque los arrays no tienen ningún objeto asociado.

Resultado:

France		
Brittany		
Brest	120000	
Quimper	80000	
Rennes	200000	
Normandy		
Caen	75000	
Deauville	4000	

Gestión optimizada de desplegar/contraer

Puede optimizar la visualización y gestión de los list box jerárquicos utilizando los eventos formulario `On Expand` y `On Collapse`.

Un list box jerárquico se construye a partir del contenido de sus arrays, por lo que sólo puede mostrarse cuando todos estos arrays están cargados en memoria. Esto dificulta la generación de list box jerárquicos de gran tamaño basados en arrays generados a partir de datos (a través del comando `SELECTION T0 ARRAY`), no sólo por la velocidad de visualización sino también por la memoria utilizada.

El uso de los eventos de formulario `On Expand` y `On Collapse` puede superar estas limitaciones: por ejemplo, puede mostrar sólo una parte de la jerarquía y cargar/descargar los arrays sobre la marcha, basándose en las acciones del usuario. En el contexto de estos eventos, el comando `LISTBOX GET CELL POSITION` devuelve la celda en la que el usuario hizo clic para desplegar o contraer una línea.

En este caso, debe llenar y vaciar los arrays por código. Los principios que deben aplicarse son:

- Cuando se muestra el list box, sólo se debe llenar el primer array. Sin embargo, debe crear un segundo array con valores vacíos para que el list box muestre los botones desplegar/contraer:

Artists/Albums/Tracks	CDs	Tracks	Durations
+ Brasil			
+ Celtic			
+ Classical			
+ Jazz			
+ New Age			
+ Others			
+ Pop/Rock			
+ Soundtrack			
+ World			

- Cuando un usuario hace clic en un botón de expandir, puede procesar el evento `On Expand`. El comando `LISTBOX GET CELL POSITION` devuelve la celda en cuestión y permite construir la jerarquía adecuada: se llena el primer array con los valores repetidos y el segundo con los valores enviados desde el comando `SELECTION TO ARRAY` y se insertan tantas líneas como sean necesarias en el list box utilizando el comando `LISTBOX INSERT ROWS`.

Artists/Albums/Tracks	CDs	Tracks	Durations
+ Brasil			
+ Celtic			
+ Classical			
+ Jazz			
+ New Age			
+ Others			
+ Jacqueline Maillan			
+ Pierre Dac			
+ Pierre Dac & Francis Blanche			
+ Pop/Rock			
+ Soundtrack			
+ World			

- Cuando un usuario hace clic en un botón de contracción, puede procesar el evento `On Collapse`. El comando `LISTBOX GET CELL POSITION` devuelve la celda en cuestión: con el comando `LISTBOX DELETE ROWS` se eliminan tantas líneas como sean necesarias del list box.

## Arrays de objetos en columnas

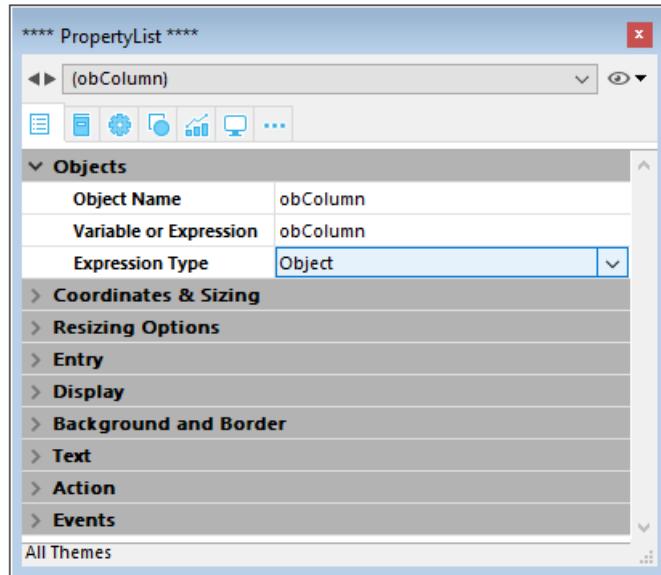
Las columnas de list box pueden manejar arrays de objetos. Como los arrays de objetos pueden contener diferentes tipos de datos, esta nueva y poderosa funcionalidad permite mezclar diferentes tipos de entrada en las líneas de una misma columna, y mostrar también varios widgets. Por ejemplo, puede insertar una entrada de texto en la primera línea, una casilla de selección en la segunda y una lista desplegable en la tercera. Los arrays de objetos también dan acceso a nuevos tipos de widgets, como botones o selectores de color.

El siguiente list box fue diseñado utilizando un array de objetos:

Label	Value
Document Name	MyReport
Document Type	PDF ▾
Reference	123456
Category	[color picker]
Include Abstract	<input checked="" type="checkbox"/>
Printable area size (height)	297 mm
Printable area size (width)	210 mm
Show Preview	Preview...

## Configuring an object array column

To assign an object array to a list box column, you just need to set the object array name in either the Property list ("Variable Name" field), or using the `LISTBOX INSERT COLUMN` command, like with any array-based column. In the Property list, you can now select Object as a "Expression Type" for the column:

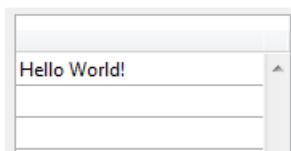


Standard properties related to coordinates, size, and style are available for object columns. You can define them using the Property list, or by programming the style, font color, background color and visibility for each row of an object-type list box column. Estos tipos de columnas también se pueden ocultar.

However, the Data Source theme is not available for object-type list box columns. In fact, the contents of each column cell are based on attributes found in the corresponding element of the object array. Cada elemento de array puede definir:

the value type (mandatory): text, color, event, etc. the value itself (optional): used for input/output. the cell content display (optional): button, list, etc. additional settings (optional): depend on the value type To define these properties, you need to set the appropriate attributes in the object (available attributes are listed below). For example, you can write "Hello World!" in an object column using this simple code:

```
ARRAY OBJECT(obColumn;0) //column array
C_OBJECT($ob) //first element
OB SET($ob;"valueType";"text") //defines the value type (mandatory)
OB SET($ob;"value";"Hello World!") //define el valor
APPEND TO ARRAY(obColumn;$ob) //define el valor
APPEND TO ARRAY(obColumn;$ob)
```



Display format and entry filters cannot be set for an object column. Dependen automáticamente del tipo de valor.

### valueType y visualización de datos

When a list box column is associated with an object array, the way a cell is displayed, entered, or edited, is based on the valueType attribute of the array element. Los valores valueType soportados son:

- "text": para un valor de texto
- "real": for a numeric value that can include separators like a <space>, <.>, o <,>
- "integer": para un valor entero
- "boolean": para un valor True/False
- "color": para definir un color de fondo

- "event": para mostrar un botón con una etiqueta.

4D uses default widgets with regards to the "valueType" value (i.e., a "text" is displayed as a text input widget, a "boolean" as a check box), but alternate displays are also available through options (e.g., a real can also be represented as a drop-down menu). The following table shows the default display as well as alternatives for each type of value:

valueType	Widget por defecto	Widget(s) alternativo(s)
texto	entrada de texto	drop-down menu (required list) or combo box (choice list)
real	entrada de texto controlada (números y separadores)	drop-down menu (required list) or combo box (choice list)
integer	entrada de texto controlada (números únicamente)	drop-down menu (required list) or combo box (choice list) or three-states check box
booleano	casilla de selección	menú desplegable (lista requerida)
color	color de fondo	texto
evento	botón con etiqueta	
		All widgets can have an additional unit toggle button or ellipsis button attached to the cell.

You set the cell display and options using specific attributes in each object (see below).

### Formatos de visualización y filtros de entrada

You cannot set display formats or entry filters for columns of object-type list boxes. They are automatically defined according to the value type. Estos están listados en la siguiente tabla:

Tipo de valor	Formato por defecto	Control de entrada
texto	lo mismo que se define en el objeto	any (no control)
real	same as defined in object (using system decimal separator)	"0-9" y "." y "-" "0-9" y "." si min>=0
integer	lo mismo que se define en el objeto	"0-9" y "-" "0-9" if min>=0
Booleano	casilla de selección	N/A
color	N/A	N/A
evento	N/A	N/A

### Atributos

Cada elemento del array de objetos es un objeto que puede contener uno o más atributos que definirán el contenido de la celda y la visualización de los datos (ver el ejemplo anterior).

The only mandatory attribute is "valueType" and its supported values are "text", "real", "integer", "boolean", "color", and "event". The following table lists all the attributes supported in list box object arrays, depending on the "valueType" value (any other attributes are ignored). Display formats are detailed and examples are provided below.

	valueType	texto	real	integer	booleano	color	evento
Atributos	Descripción						
value	valor de la celda (entrada o salida)	x	x	x			
min	valor mínimo		x	x			
max	valor máximo		x	x			
behavior	Valor "tres Estados"			x			
requiredList	lista desplegable definida en objeto	x	x	x			
choiceList	combo box definido en objeto	x	x	x			
requiredListReference	4D list ref, depends on "saveAs" value	x	x	x			
requiredListName	Nombre de la lista 4D, depende del valor "saveAs"	x	x	x			
saveAs	"reference" o "value"	x	x	x			
choiceListReference	4D list ref, display combo box	x	x	x			
choiceListName	Nombre de la lista 4D, mostrar combo box	x	x	x			
unitList	array de X elementos	x	x	x			
unitReference	índice del elemento seleccionado	x	x	x			
unitsListReference	Ver lista de unidades 4D	x	x	x			
unitsListName	4D lista nombre de la unidad	x	x	x			
alternateButton	añadir un botón alternativo	x	x	x	x	x	

### value

Los valores de las celdas se almacenan en el atributo "valor". This attribute is used for input as well as output. También puede utilizarse para definir valores por defecto cuando se utilizan listas (ver a continuación).

```

ARRAY OBJECT(obColumn;0) //array columna
C_OBJECT($obj1)
$entry:="Hello world!"
ARRAY OBJECT(obColumn;0) //array columna
C_OBJECT($obj1)
$entry:="Hello world!"
ARRAY OBJECT(obColumn;0) //array columna
C_OBJECT($obj1)
$entry:="Hello world!"
OB SET($obj1;"valueType";"text")
OB SET($obj1;"value";$entry) // si el usuario introduce un nuevo valor, $entry contendrá el valor editad
C_OBJECT($obj2)
OB SET($obj2;"valueType";"real")
OB SET($obj2;"value";2/3)
C_OBJECT($obj3)
OB SET($obj3;"valueType";"boolean")
OB SET($obj3;"value";True)

APPEND TO ARRAY(obColumn;$obj1)
APPEND TO ARRAY(obColumn;$obj2)
APPEND TO ARRAY(obColumn;$obj3)
ARRAY OBJECT(obColumn;0) //array columna
C_OBJECT($obj1)
$entry:="Hello world!"
ARRAY OBJECT(obColumn;0) //array columna
C_OBJECT($obj1)
$entry:="Hello world!"
ARRAY OBJECT(obColumn;0) //array columna
C_OBJECT($obj1)
$entry:="Hello world!"
OB SET($obj1;"valueType";"text")
OB SET($obj1;"value";$entry) // si el usuario introduce un nuevo valor, $entry contendrá el valor editad
C_OBJECT($obj2)
OB SET($obj2;"valueType";"real")
OB SET($obj2;"value";2/3)
C_OBJECT($obj3)
OB SET($obj3;"valueType";"boolean")
OB SET($obj3;"value";True)

APPEND TO ARRAY(obColumn;$obj1)
APPEND TO ARRAY(obColumn;$obj2)
APPEND TO ARRAY(obColumn;$obj3)
ARRAY OBJECT(obColumn;0) //array columna
C_OBJECT($obj1)
$entry:="Hello world!"
ARRAY OBJECT(obColumn;0) //array columna
C_OBJECT($obj1)
$entry:="Hello world!"
OB SET($obj1;"valueType";"text")
OB SET($obj1;"value";$entry) // si el usuario introduce un nuevo valor, $entry contendrá el valor editad
C_OBJECT($obj2)
OB SET($obj2;"valueType";"real")
OB SET($obj2;"value";2/3)
C_OBJECT($obj3)
OB SET($obj3;"valueType";"boolean")
OB SET($obj3;"value";True)

APPEND TO ARRAY(obColumn;$obj1)
APPEND TO ARRAY(obColumn;$obj2)
APPEND TO ARRAY(obColumn;$obj3)

```

Header1
Hello world!
0,66666666666667
<input checked="" type="checkbox"/>

Null values are supported and result in an empty cell.

## min y max

Cuando el "valueType" es "real" o "integer", el objeto también acepta atributos min y max con valores apropiados (los valores deben ser del mismo tipo que el valueType).

Estos atributos pueden utilizarse para controlar el rango de valores de entrada. Cuando se valida una celda (cuando pierde el foco), si el valor de entrada es menor que el valor mínimo o mayor que el valor máximo, entonces se rechaza. En este caso, se mantiene el valor anterior y un consejo muestra una explicación.

```
C_OBJECT($ob3)
$entry3:=2015
OB SET($ob3;"valueType";"integer")
OB SET($ob3;"value";$entry3)
OB SET($ob3;"min";2000)
OB SET($ob3;"max";3000)
```

## behavior

El atributo behavior ofrece variaciones a la representación estándar de los valores. En 4D v15, se ofrece una única variación:

Atributo	Valor(es) disponible(s)	valueType(s)	Descripción
behavior	threeStates	integer	Representa un valor numérico como una casilla de selección de tres estados. 2=intermediario, 1=seleccionado, 0=no seleccionado, -1=invisible, -2=no seleccionado desactivado, -3=seleccionado desactivado, -4=semi seleccionado desactivado

```
C_OBJECT($ob3)
OB SET($ob3;"valueType";"integer")

OB SET($ob3;"value";-3)
C_OBJECT($ob4)
OB SET($ob4;"valueType";"integer")
OB SET($ob4;"value";-3)
OB SET($ob4;"behavior";"threeStates")
```

## requiredList y choiceList

Cuando un atributo "choiceList" o "requiredList" está presente dentro del objeto, la entrada de texto se sustituye por una lista desplegable o un combo box, dependiendo del atributo:

- Si el atributo es "choiceList", la celda se muestra como un combo box. Esto significa que el usuario puede seleccionar o escribir un valor.
- If the attribute is "requiredList" then the cell is displayed as a drop-down list and the user can only select one of the values provided in the list.

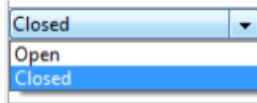
In both cases, a "value" attribute can be used to preselect a value in the widget.

Los valores del widget se definen a través de un array. Si quiere asociar el widget a una lista 4D existente, debe utilizar los atributos "requiredListReference", "requiredListName", "choiceListReference" o "choiceListName".

Ejemplos:

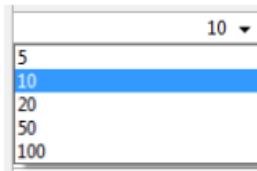
- Quiere mostrar una lista desplegable con sólo dos opciones: "Open" o "Closed". "Closed" debe estar preseleccionado:

```
ARRAY TEXT($RequiredList;0)
APPEND TO ARRAY($RequiredList;"Open")
APPEND TO ARRAY($RequiredList;"Closed")
C_OBJECT($ob)
OB SET($ob;"valueType";"text")
OB SET($ob;"value";"Closed")
OB SET ARRAY($ob;"requiredList";$RequiredList)
```



- Quiere aceptar todo valor entero, pero mostrar un combo box para sugerir los valores más comunes:

```
ARRAY LONGINT($ChoiceList;0)
APPEND TO ARRAY($ChoiceList;5)
APPEND TO ARRAY($ChoiceList;10)
APPEND TO ARRAY($ChoiceList;20)
APPEND TO ARRAY($ChoiceList;50)
APPEND TO ARRAY($ChoiceList;100)
C_OBJECT($ob)
OB SET($ob;"valueType";"integer")
OB SET($ob;"value";10) //10 como valor por defecto
OB SET ARRAY($ob;"choiceList";$ChoiceList)
```



requiredListName y requiredListReference

Los atributos "requiredListName" y "requiredListReference" permiten utilizar, en una celda de list box, una lista definida en 4D, ya sea en modo Diseño (en el editor de Listas de la Caja de Herramientas) o por programación (utilizando el comando New list). La celda se mostrará entonces como una lista desplegable. Esto significa que el usuario sólo puede seleccionar uno de los valores proporcionados en la lista.

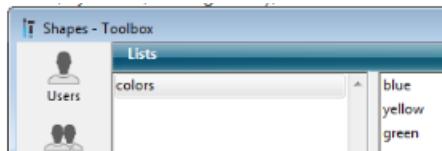
Utilice "requiredListName" o "requiredListReference" en función del origen de la lista: si la lista procede de la caja de herramientas, pase un nombre; en caso contrario, si la lista se ha definido por programación, pase una referencia. In both cases, a "value" attribute can be used to preselect a value in the widget.

- Si desea definir estos valores a través de un simple array, debe utilizar el atributo "requiredList".
- Si la lista contiene elementos de texto que representan valores reales, el separador decimal debe ser un

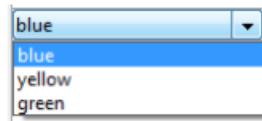
punto ("."), independientemente de la configuración local, por ejemplo "17.6" "1234.456".

#### Ejemplos:

- Desea mostrar una lista desplegable basada en una lista de "colores" definida en la caja de herramientas (que contiene los valores "azul", "amarillo" y "verde"), guardarla como valor y mostrar "azul" por defecto:



```
C_OBJECT($ob)
OB SET($ob;"valueType";"text")
OB SET($ob;"saveAs";"value")
OB SET($ob;"value";"blue")
OB SET($ob;"requiredListName";"colors")
```



- Quiere mostrar una lista desplegable basada en una lista definida por programación y guardarla como referencia:

```
<>List:=New list
APPEND TO LIST(<>List;"Paris";1)
APPEND TO LIST(<>List;"London";2)
APPEND TO LIST(<>List;"Berlin";3)
APPEND TO LIST(<>List;"Madrid";4)
C_OBJECT($ob)
OB SET($ob;"valueType";"integer")
OB SET($ob;"saveAs";"reference")
OB SET($ob;"value";2) //muestra Londres por defecto
OB SET($ob;"requiredListReference";<>List)
```

! [] (/docs/Rx/assets/en/FormObjects/listbox\_column\_objectArray\_cities.png)

#### choiceListName y choiceListReference

Los atributos "choiceListName" and "choiceListReference" permiten utilizar, en una celda de list box, una lista definida en 4D, ya sea en modo Diseño (en el editor de Listas de la Caja de Herramientas) o por programación (utilizando el comando New list). La celda se muestra entonces como un combo box, lo que significa que el usuario puede seleccionar o escribir un valor.

Utilice "choiceListName" o "choiceListReference" en función del origen de la lista: si la lista procede de la caja de herramientas, pase un nombre; en caso contrario, si la lista se ha definido por programación, pase una referencia. In both cases, a "value" attribute can be used to preselect a value in the widget.

- Si desea definir estos valores a través de un simple array, debe utilizar el atributo "choiceList".
- Si la lista contiene elementos de texto que representan valores reales, el separador decimal debe ser un punto ("."), independientemente de la configuración local, por ejemplo "17.6" "1234.456".

#### Ejemplo:

Ejemplo:



```
C_OBJECT($ob)
OB SET($ob;"valueType";"text")
OB SET($ob;"value";"blue")
OB SET($ob;"choiceListName";"colors")
```

unitsList, unitsListName, unitsListReference y unitReference

Puede utilizar atributos específicos para añadir unidades asociadas a los valores de las celdas (\*por ejemplo, \*: "10 cm", "20 píxeles", etc.). Para definir la lista de unidades, puede utilizar uno de los siguientes atributos:

- "unitsList": un array que contiene los elementos x utilizados para definir las unidades disponibles (por ejemplo: "cm", "pulgadas", "km", "millas", etc.). Utilice este atributo para definir las unidades dentro del objeto.
- "unitsListReference": una referencia a una lista 4D que contiene las unidades disponibles. Utilice este atributo para definir unidades con una lista 4D creada con el comando [New list](#).
- "unitsListName": un nombre de una lista 4D basada en el diseño que contiene unidades disponibles. Utilice este atributo para definir las unidades con una lista 4D creada en la caja de herramientas.

Independientemente de la forma en que se defina la lista de unidades, puede asociarse con el siguiente atributo:

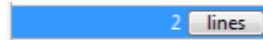
- "unitReference": un único valor que contiene el índice (de 1 a x) del elemento seleccionado en la lista de valores "unitList", "unitsListReference" o "unitsListName".

Independientemente de la forma en que se defina la lista de unidades, puede asociarse con el siguiente atributo:

Ejemplo:

Queremos definir una entrada numérica seguida de dos posibles unidades: " líneas " o " píxeles ". El valor actual es "2" + "líneas". Utilizamos valores definidos directamente en el objeto (atributo "unitsList"):

```
ARRAY TEXT($_units;0)
APPEND TO ARRAY($_units;"lines")
APPEND TO ARRAY($_units;"pixels")
C_OBJECT($ob)
OB SET($ob;"valueType";"integer")
OB SET($ob;"value";2) // 2 "units"
OB SET($ob;"unitReference";1) //"lines"
OB SET ARRAY($ob;"unitsList";$_units)
```



alternateButton

Si desea añadir un botón de elipsis [...] a una celda, sólo tiene que pasar el atributo "alternateButton" con el valor True en el objeto. El botón se mostrará en la celda automáticamente.

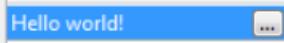
Cuando este botón es presionado por un usuario, se generará un evento `On Alternate Click`, y usted podrá manejarlo como quiera (vea el párrafo "Manejo de eventos" para más información).

Ejemplo:

```

C_OBJECT($ob1)
$entry:="Hello world!"
C_OBJECT($ob1)
$entry:="Hello world!"
OB SET($ob;"valueType";"text")
OB SET($ob;"alternateButton";True)
OB SET($ob;"value";$entry)
C_OBJECT($ob1)
$entry:="Hello world!"
OB SET($ob;"valueType";"text")
OB SET($ob;"alternateButton";True)
OB SET($ob;"value";$entry)
C_OBJECT($ob1)
$entry:="Hello world!"
OB SET($ob;"valueType";"text")
OB SET($ob;"alternateButton";True)
OB SET($ob;"value";$entry)
C_OBJECT($ob1)
$entry:="Hello world!"
OB SET($ob;"valueType";"text")
OB SET($ob;"alternateButton";True)
OB SET($ob;"value";$entry)

```



## valueType color

El atributo valueType de valor "color" permite mostrar un color o un texto.

- Si el valor es un número, se dibuja un rectángulo de color dentro de la celda. Ejemplo:

```

C_OBJECT($ob4)
OB SET($ob4;"valueType";"color")
OB SET($ob4;"value";0x00FF0000)

```



- If the value is a text, then the text is displayed (e.g.: "value";"Automatic").

## event valueType

The "event" valueType displays a simple button that generates an On Clicked event when clicked. No se puede pasar ni devolver ningún dato o valor.

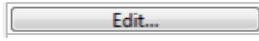
Opcionalmente, se puede pasar un atributo "label".

Ejemplo:

```

C_OBJECT($ob)
OB SET($ob;"valueType";"event")
OB SET($ob;"label";"Edit...")

```



## Gestión de eventos

Several events can be handled while using an object list box array:

- On Data Change: An `On Data Change` event is triggered when any value has been modified either:
  - en un área de entrada de texto
  - en una lista desplegable
  - en un área combo box
  - in a unit button (switch from value x to value x+1)
  - en una casilla de selección (cambia entre marcado/desmarcado)
- On Clicked: When the user clicks on a button installed using the "event" `valueType` attribute, an `On Clicked` event will be generated. Este evento es gestionado por el programador.
- On Alternative Click: When the user clicks on an ellipsis button ("alternateButton" attribute), an `On Alternative Click` event will be generated. Este evento es gestionado por el programador.

# Botón Imagen

Un botón imagen es similar a un [botón estándar](#). Sin embargo, a diferencia de un botón estándar (que acepta tres estados: activado, desactivado y pulsado), un botón imagen tiene una imagen diferente para representar cada estado.

Los botones imagen pueden utilizarse de dos maneras:

- Como botones de comando en un formulario. En este caso, el botón imagen generalmente incluye cuatro estados diferentes: activado, desactivado, presionado y pasado por encima.

Por ejemplo, una tabla de miniaturas que tiene una línea de cuatro columnas, cada una de las miniaturas corresponde a los estados Por defecto, Presionado, Pasado por encima y Desactivado.

Propiedad	Nombre JSON	Valor
Rows	rowCount	1
Columnas	columnCount	4
Switch back when Released	switchBackWhenReleased	true
Switch when Roll Over	switchWhenRollover	true
Utilizar el Último cuadro como Desactivado	useLastFrameAsDisabled	true

- Como botón de imagen que permite al usuario elegir entre varias opciones. En este caso, se puede utilizar un botón de imagen en lugar de un menú de imagen emergente. Con los [Menús emergentes imagen](#), todas las opciones se muestran simultáneamente (como los elementos del menú emergente), mientras que un botón imagen muestra las opciones consecutivamente (a medida que el usuario hace clic en el botón).

Este es un ejemplo de botón imagen. Supongamos que quiere dar a los usuarios de una aplicación personalizada la posibilidad de elegir el idioma de la interfaz de la aplicación. La opción se implementa como un botón imagen en una caja de diálogo personalizada de propiedades:



Al hacer clic en el objeto, la imagen cambia.

## Utilizar los botones imagen

Puede implementar un botón imagen de la siguiente manera:

1. En primer lugar, prepare un único gráfico en el que las series de imágenes estén dispuestas en líneas, en columnas o en las dos.



Puede organizar las imágenes en columnas, líneas o en una cuadrícula (como se muestra arriba). Cuando se organizan las imágenes en forma de cuadrícula, se numeran de izquierda a derecha, línea por línea, empezando por 0. Por ejemplo, la segunda imagen de la segunda línea de una cuadrícula que consta de dos líneas y tres columnas, tiene el número 4 (la bandera del Reino Unido en el ejemplo anterior).

2. A continuación, asegúrese de que la imagen está en los recursos de su proyecto e introduzca la ruta en la propiedad [Ruta de acceso imagen](#).
3. Defina las propiedades de [líneas y columnas](#) del gráfico.

4. Especifique cuándo cambian las imágenes seleccionando las propiedades de [animación](#) apropiadas.

## Animación

Además de los parámetros de posicionamiento y de apariencia estándar, puede definir algunas propiedades específicas para los botones imagen, especialmente en lo que respecta a cómo y cuándo se muestran las imágenes. Estas opciones de propiedades pueden combinarse para mejorar sus botones de imagen.

- Por defecto ([cuando no se selecciona la opción animación](#)), un botón de imagen muestra la siguiente imagen de la serie cuando el usuario hace clic; muestra la imagen anterior de la serie cuando el usuario mantiene pulsada la tecla Mayúsculas y hace clic en el botón. Cuando el usuario llega a la última imagen de la serie, la imagen no cambia cuando el usuario hace clic de nuevo. En otras palabras, no vuelve a la primera imagen de la serie.

Hay otros modos disponibles:

- [Vuelve a la primera secuencia](#)
- [Switch back when Released](#)
- [Switch when Roll Over](#)
- [Desplazamiento continuo en clics](#)
- [Utilizar el Último cuadro como Desactivado](#)
- [Utilizar el Último cuadro como Desactivado](#)

[Utilizar el último cuadro como desactivado](#) > [Utilizar el último cuadro como desactivado](#) > [Utilizar el último cuadro como desactivado](#) > La [variable asociada](#) al botón imagen devuelve el número de índice, en la tabla de miniaturas, de la imagen actual mostrada. La numeración de las imágenes en la tabla empieza por 0.

## Propiedades soportadas

Negrita - Estilo del borde - Abajo - Estilo del botón - Class - Columnas - Soltable - Focusable - Fuente - Color de la fuente - Altura - Mensaje de ayuda - Dim. Horizontal - Itálica - Izquierda - Recomenzar la secuencia - Nombre - Ruta de acceso de la imagen - Derecha - Líneas - Atajo - Acción estándar - Vuelve a cambiar cuando se libera - Desplazamiento continuo en clic - Desplazamiento cada x ticks - Título - Recomenzar la secuencia - Arriba - Type - Use Last frame as disabled - Variable o expresión - Dim. vertical - Visibilidad - Ancho

# Menú pop-up imagen

Un menú emergente de imágenes que muestra un array de imágenes en dos dimensiones. Se puede utilizar un menú emergente de imágenes en lugar de un [botón imagen](#). La creación de la imagen a utilizar con un menú emergente imagen es similar a la creación de una imagen para un botón imagen. El concepto es el mismo que el de la [rejillas de botones](#), salvo que la imagen se utiliza como un menú emergente en lugar de un objeto del formulario.

## Utilizar los menús emergentes de imágenes

Para crear un menú emergente de imágenes, debe [referirse a una imagen](#). El siguiente ejemplo permite seleccionar el idioma de la interfaz seleccionandolo en un menú emergente de imágenes. Cada idioma está representado por la bandera correspondiente:



## Programación

Puede gestionar los menús emergentes de imágenes utilizando métodos. Al igual que con las [rejillas de botones](#), las variables asociadas a los menús emergentes de imágenes se definen con el valor del elemento seleccionado en el menú emergente de imágenes. Si no se selecciona ningún elemento, el valor es 0. Los elementos están numerados, línea por línea, de izquierda a derecha, empezando por la línea superior.

## Ir a la página

Puede asociar el `gotoPage` acción estándar a un objeto de tipo pop-up menu imagen. Cuando se selecciona esa acción, 4D mostrará automáticamente la página del formulario que corresponde a la posición de la imagen seleccionada en el array de imágenes. Los elementos se numeran de izquierda a derecha y de arriba a abajo, empezando por la esquina superior izquierda.

Por ejemplo, si el usuario selecciona el tercer elemento, 4D mostrará la página 3 del formulario actual (si existe). Si desea gestionar usted mismo el efecto de un clic, seleccione Sin acción .

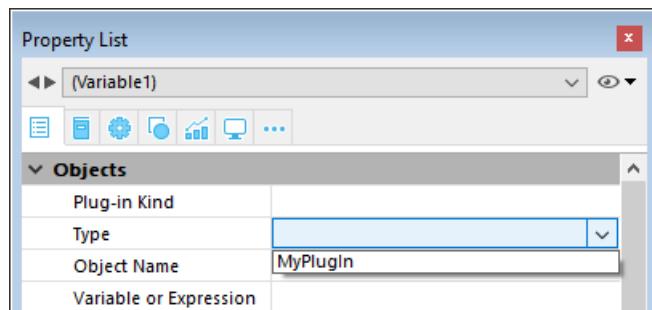
## Propiedades soportadas

[Negrita](#) - [Estilo del borde](#) - [Inferior](properties\_CoordinatesAndSizing. md#bottom) - [Clase](#) - [Columnas](#) - [Altura](#) - [Consejo de ayuda](#) - [Tamaño horizontal](properties\_ResizingOptions. md#horizontal-sizing) - [Izquierda](#) - [Nombre del objeto](properties\_Object. md#object-name) - [Ruta de acceso](#) - [Derecha](#) - [Filas](#) - [Acción estándar](properties\_Action. md#standard-action) - [Superior](#) - [Tipo](properties\_Object. md#type) - [Variable o expresión](#) - [Tamaño vertical](properties\_ResizingOptions. md#vertical-sizing) - [Visibilidad](#) - [Ancho](#)

# Área de plug-in

Un área de plug-in es un área en el formulario que está completamente controlada por un plug-in. La capacidad de integrar plug-ins en los formularios le ofrece posibilidades ilimitadas a la hora de crear aplicaciones personalizadas. Un plug-in puede realizar una tarea sencilla, como mostrar un reloj digital en un formulario, o una tarea compleja, como ofrecer funciones completas de procesamiento de textos, hojas de cálculo o gráficos.

Al abrir una aplicación, 4D crea una lista interna de los plug-ins [instalados en la aplicación](#). Una vez que haya insertado un área de plug-in en un formulario, puede asignar un plug-in al área directamente en la lista Tipo en la lista de propiedades:



Ciertos plug-ins, como 4D Internet Commands, no pueden utilizarse en formularios o ventanas externas. Cuando un plug-in no puede ser utilizado en un formulario, no aparece en la lista de plug-ins de la Lista de Propiedades.

Si dibuja un área de plug-in demasiado pequeña, 4D la mostrará como un botón cuyo título es el nombre de la variable asociada al área. Durante la ejecución, el usuario puede hacer clic en este botón para abrir una ventana específica que muestre el plug-in.

## Propiedades avanzadas

Si el autor del plugin ofrece opciones avanzadas, se puede habilitar un tema Plug-in que contenga un botón [Propiedades avanzadas](#) en la lista de propiedades. En este caso, puede hacer clic en este botón para definir estas opciones, normalmente a través de una caja de diálogo personalizada.

## Instalar un plug-in

Para añadir un plug-in en tu entorno 4D, primero tiene que salir de 4D. Los plug-ins se cargan al iniciar 4D. Para más información sobre la instalación de plug-ins, consulte [Instalación de plug-ins o componentes](#).

## Crear plug-ins

Si está interesado en diseñar sus propios plug-ins, puede recibir amplia información sobre cómo escribir e implementar plug-ins. 4D ofrece un [kit completo \(en github\)](#) para ayudarle a escribir plug-ins personalizados.

## Propiedades soportadas

[Estilo del borde](#) - [Abajo](#) - [Propiedades avanzadas](#) - [Class](#) - [Arrastrable](#) - [Soltable](#) - [Tipo de expresión](#) - [Focusable](#) - [Altura](#) - [Dim. horizontal](#) - [Izquierda](#) - [Método](#) - [Nombre de objeto](#) - [Tipo de Plug-in](#) - [Derecha](#) - [Arriba](#) - [Tipo](#) - [Variable o expresión](#) - [Dim. vertical](#) - [Visibilidad](#) - [Ancho](#)

# Indicador de progreso

Un indicador de progreso (también llamado "termómetro") está diseñado para mostrar o definir gráficamente los valores numéricos o fecha/hora.



## Utilizar los indicadores

Los indicadores se pueden utilizar tanto para visualizar como para definir valores. Por ejemplo, si a un indicador de progreso se le da un valor por un método, muestra el valor. Si el usuario arrastra el punto indicador, el valor cambia. El valor puede utilizarse en otro objeto, como un campo o un objeto introducible o no introducible.

La variable asociada al indicador controla la visualización. Puede introducir y utilizar los valores del indicador utilizando métodos. Por ejemplo, un método para un campo o un objeto introducible podría utilizarse para controlar un indicador de progreso:

```
vTherm:=[Employees]Salary
```

Este método asigna el valor del campo Salary a la variable vTherm. Este método se adjuntará al campo Salario.

Por el contrario, puede utilizar el indicador para controlar el valor de un campo. El usuario arrastra el indicador para definir el valor. En este caso el método se convierte en:

```
[Employees]Salary:=vTherm
```

El método asigna el valor del indicador al campo Salario. A medida que el usuario arrastra el indicador, el valor del campo Salario cambia.

## El termómetro por defecto



El termómetro es el indicador básico de progreso.

Puede mostrar barras de termómetros horizontales o verticales. Esto viene determinado por la forma del objeto que se dibuja.

Dispone de múltiples opciones gráficas: valores mínimos/máximos, graduaciones, pasos.

## Propiedades soportadas

Barber shop - Negrita - Estilo de borde - Abajo - Clase - Mostrar graduación - Ingresable - Ejecutar método objeto - Tipo de expresión (sólo "integer", "number", "date", o "time") - Fuente - Color de fuente - Tamaño de fuente - Altura - Itálica - Unidad de graduación - Mensaje de ayuda - Dimensionamiento horizontal - Ubicación etiqueta - Izquierda - Máximo - Mínimo - Formato numérico - Nombre objeto - Derecho - Paso - Arriba - Tipo - Subrayado - Variable o expresión - Dimensionamiento vertical - Visibilidad - Ancho

# Barber shop



Barber shop es una variante del termómetro por defecto. Para activar esta variante, es necesario definir la propiedad **Barber shop**.

En JSON, basta con eliminar la propiedad "max" del objeto termómetro por defecto para activar la variante Barber shop.

La Barber shop muestra una animación continua, como la [spinner](#). Estos termómetros se utilizan generalmente para indicar al usuario que el programa está en proceso de realizar una operación larga. Cuando se selecciona esta variante termómetro, [las propiedades de la escala gráfica](#) no están disponibles.

Cuando se ejecuta el formulario, el objeto no se anima. La animación se gestiona pasando un valor a su [variable o expresión asociada](#):

- 1 (o cualquier valor diferente de 0) = Iniciar la animación,
- 0 = Detener la animación.

## Propiedades soportadas

Barber shop - Negrita - Estilo de borde - [Abajo](#) - Clase - Ingresable - Ejecutar método objeto - Tipo de expresión (sólo "integer", "number", "date" o "time") - Fuente - Color de fuente - Tamaño de fuente - Altura - Mensaje de ayuda - Dimensionamiento horizontal - Itálica - Izquierda - Nombre del objeto - Derecha - Arriba - Tipo - Subrayado - Variable o expresión - Dimensionamiento vertical - Visibilidad - Ancho

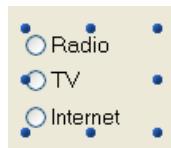
## Ver también

- [reglas](#)
- [steppers](#)

# Botón radio

Los botones radio son objetos que permiten al usuario seleccionar uno de un grupo de botones.

Normalmente, un botón radio muestra una pequeña diana con texto. Sin embargo, los botones radio pueden tener diferentes apariencias.



Se selecciona un botón radio:

- cuando el usuario hace clic en él
- cuando tiene el foco y el usuario presiona la tecla Barra espaciadora.

## Configuración de botones radio

Los botones radio se utilizan en conjuntos coordinados: sólo se puede seleccionar un botón a la vez en el conjunto. Para funcionar de forma coordinada, un conjunto de botones radio debe compartir la misma propiedad **Grupo radio**.

Los botones radio se controlan con métodos. Como todos los botones, la variable asociada al botón radio se inicializa en 0 cuando se abre el formulario por primera vez. Un método asociado a un botón radio se ejecuta cuando se selecciona el botón. A continuación se muestra un ejemplo de un grupo de botones radio utilizados en una base de datos de registro de vídeos para introducir la velocidad del registro (SP, LP o EP):



Al seleccionar un botón radio de un grupo, ese botón se pone en 1 y todos los demás del grupo en 0. Sólo se puede seleccionar un botón de radio a la vez.

Puede asociar [expresiones de tipo booleano](#) a botones radio. En este caso, cuando se selecciona un botón radio de un grupo, su variable es True y las variables de los demás botones radio del grupo son False.

El valor contenido en un objeto botón radio no se guarda automáticamente (excepto si es la representación de un campo booleano); los valores de los botones radio deben almacenarse en sus variables y gestionarse con métodos.

## Estilos de botón

[Los estilos de botón](#) controlan la apariencia general del botón de radio y sus propiedades disponibles. Es posible aplicar diferentes estilos predefinidos a los botones radio. Sin embargo, debe aplicarse el mismo estilo de botón a todos los botones de radio de un grupo para que funcionen como se espera.

4D ofrece botones radio en los siguientes estilos predefinidos:

### Clásico

El estilo de botón radio Clásico es un botón sistema estándar (\*es decir, \*, una pequeña diana con texto) que ejecuta código cuando el usuario hace clic en él.



Además de iniciar la ejecución del código, el estilo del botón radio Clásico cambia el color de esfera cuando se pasa por encima.

## Plano

El estilo de botón radio Plano es un botón sistema estándar (\*es decir, \*, una pequeña diana con texto) que ejecuta código cuando el usuario hace clic en él.



Por defecto, el estilo Plano tiene una apariencia minimalista. El estilo gráfico del botón Flat es especialmente útil para los formularios a imprimir.

## Barra de herramientas

El estilo del botón radio Barra de herramientas está pensado principalmente para su integración en una barra de herramientas.

Por defecto, el estilo Barra de herramientas tiene un fondo transparente con una etiqueta en el centro. La apariencia del botón puede ser diferente cuando el cursor pasa por encima de él dependiendo del sistema operativo:

- *Windows* - el botón está resaltado.

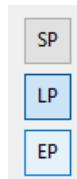


- *macOS* - el resalte del botón nunca aparece.

## Bevel

El estilo de botón radio Bevel es similar al comportamiento del estilo [Barra de herramientas](#), excepto que tiene un fondo gris claro y un contorno gris. La apariencia del botón puede ser diferente cuando el cursor pasa por encima de él dependiendo del sistema operativo:

- *Windows* - el botón está resaltado.



- *macOS* - el resalte del botón nunca aparece.

## Bevel redondeado

El estilo de botón Bevel redondeado es casi idéntico al estilo [Bevel](#), excepto que, dependiendo del sistema operativo, las esquinas del botón pueden ser redondeadas.

- *Windows* - el botón es idéntico al estilo [Bevel](#).
- *macOS* - las esquinas del botón están redondeadas.



## OS X Gradient

El estilol botón OS X Gradient es casi idéntico al estilo [Bevel](#), excepto que, dependiendo del sistema operativo, puede tener una apariencia de dos tonos.

- *Windows* - el botón es idéntico al estilo [Bevel](#).
- *macOS* - el botón se muestra como un botón de dos tonos.

## OS X Texturizado

El estilo del botón radio OS X Textured es casi idéntico al estilo [Barra de herramientas](#) excepto que, dependiendo del sistema operativo, puede tener una apariencia diferente y no mostrar cuando el cursor pasa por encima.

Por defecto, el estilo OS X Textured aparece como:

- *Windows* -un botón en forma de barra de herramientas con una etiqueta en el centro y el fondo se muestra siempre.
- *macOS* - - un botón sistema estándar que muestra un cambio de color de gris claro a gris oscuro. Su altura está predefinida: no es posible ampliarla o reducirla.

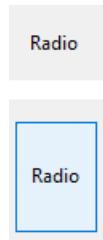


## Office XP

The Office XP button style combines the appearance of the [Regular](#) style (standard system button) with the [Toolbar](#) style's behavior.

Los colores (resaltado y fondo) de un botón con el estilo Office XP se basan en los colores del sistema. La apariencia del botón puede ser diferente cuando el cursor pasa por encima de él dependiendo del sistema operativo:

- *Windows* - su fondo sólo aparece cuando el ratón pasa por encima.



- *macOS* - su fondo se muestra siempre.

## Contraer/Desplegar

Este estilo de botón se puede utilizar para añadir un ícono estándar contraer/desplegar. Estos botones se utilizan de forma nativa en las listas jerárquicas. Estos botones se utilizan de forma nativa en las listas jerárquicas.



## Botón de divulgación

El estilo de botón radio de divulgación muestra el botón radio como un botón de divulgación estándar, normalmente utilizado para mostrar/ocultar información adicional. El símbolo del botón apunta hacia abajo con el valor 0 y hacia arriba con el valor 1.



## Personalizado

El estilo de botón radio Personalizado acepta una imagen de fondo personalizada y permite gestionar parámetros adicionales como [desplazamiento del icono](#) y las [márgenes](#).

## Propiedades soportadas

Todos los botones radio comparten el mismo conjunto de propiedades básicas:

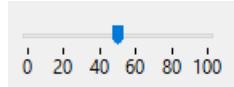
[Bold](#) - [Bottom](#) - [Button Style](#) - [Class](#) - [Expression Type](#) - [Focusable](#) - [Font](#) - [Font Color](#) - [Height](#) - [Help Tip](#) - [Horizontal Sizing](#) - [Italic](#) - [Left](#) - [Method](#) - [Object Name](#) - [Radio Group](#) - [Right](#) - [Save value](#) - [Shortcut](#) - [Title](#) - [Top](#) - [Type](#) - [Underline](#) - [Variable or Expression](#) - [Vertical Sizing](#) - [Visibility](#) - [Width](#)

Propiedades específicas adicionales están disponibles dependiendo del [estilo-de-botón](#):

- [Ruta de acceso fondo](#) - [Margen horizontal](#) - [Desplazamiento icono](#) - [Margen vertical](#) (Personalizado)
- [Número de estados](#) - [Ruta de acceso imagen](#) - [Posición Título/Imagen](#) (Botón barra de herramientas, Bevel Redondeado, OS X Gradient, OS X Textured, Office XP, Personalizado)

# Regla

The ruler is a standard interface object used to set or get values using a cursor moved along its graduations.



You can assign its [associated variable or expression](#) to an enterable area (field or variable) to store or modify the current value of the object.

For more information, please refer to [Using indicators](#) in the "Progress Indicator" page.

## Propiedades soportadas

[Bold](#) - [Border Line Style](#) - [Bottom](#) - [Class](#) - [Display graduation](#) - [Enterable](#) - [Execute object method](#) - [Expression Type](#) - [Height](#) - [Graduation step](#) - [Help Tip](#) - [Horizontal Sizing](#) - [Label Location](#) - [Left](#) - [Maximum](#) - [Minimum](#) - [Number Format](#) - [Object Name](#) - [Right](#) - [Step](#) - [Top](#) - [Type](#) - [Variable or Expression](#) - [Vertical Sizing](#) - [Visibility](#) - [Width](#)

## Ver también

- [indicadores de progreso](#)
- [steppers](#)

# Formas

Las formas son [objetos estáticos](#) que pueden añadirse a los formularios 4D.

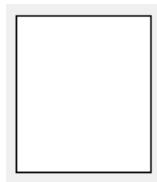
Se ofrecen las siguientes formas básicas:

- rectángulos
- líneas
- óvalos

## Rectángulo

Un rectángulo estático es un objeto decorativo para los formularios. Los rectángulos se limitan a formas cuadradas.

El diseño de los rectángulos se controla a través de muchas propiedades (color, grosor de línea, patrón, etc.). En concreto, se puede definir la [redondez](#) de sus esquinas.



Ejemplo JSON:

```
"myRectangle": {  
    "type": "rectangle",      //define el tipo de objeto  
    "left": 60,                //posición izquierda en el formulario  
    "top": 160,                //posición superior en el formulario  
    "width": 100,               //ancho del objeto  
    "height": 20,                //altura del objeto  
    "borderRadius": 20        //definir la redondez de las esquinas  
}
```

Propiedades soportadas

Abajo - Clase Css - Radio de la esquina - Tipo de línea punteada - Color de fondo - Altura - Dimensionamiento horizontal - Izquierda - Color de línea - Ancho de línea - Nombre del objeto - Derecha - Arriba - Tipo - Dimensionamiento vertical - Visibilidad - Ancho

## Línea

Una línea estática es un objeto decorativo para los formularios, trazado entre dos tramas. Las líneas pueden ser horizontales, verticales o de cualquier forma de ángulo.

El diseño de las líneas se controla a través de muchas propiedades (color, grosor de línea, etc.).

### propiedad startPoint

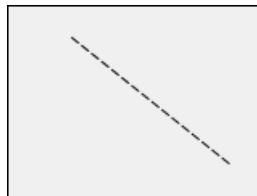
La propiedad JSON `startPoint` define a partir de qué coordenada dibujar la línea (ver ejemplo).

la propiedad `startPoint` no está expuesta en la lista de propiedades, donde la dirección de dibujo de la línea es visible.

Ejemplos JSON:

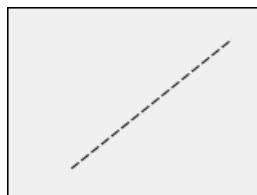
```
"myLine": {  
    "type": "line",  
    "left": 20,  
    "top": 40,  
    "width": 100,  
    "height": 80,  
    "startPoint": "topLeft", //primera orientación  
    "strokeDashArray": "6 2" //punteado  
}
```

Resultado:



```
"myLine": {  
    "type": "line",  
    "left": 20,  
    "top": 40,  
    "width": 100,  
    "height": 80,  
    "startPoint": "bottomLeft", //segunda orientación  
    "strokeDashArray": "6 2" //dashed  
}
```

Resultado:

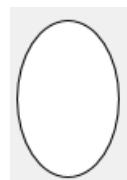


Propiedades soportadas

[Abajo](#) - [Clase Css](#) - [Tipo de línea punteada](#) - [Altura](#) - [Dimensionamiento horizontal](#) - [Izquierda](#) - [Color de línea](#) - [Ancho de línea](#) - [Nombre del objeto](#) - [Derecha](#) - [startPoint](#) - [Arriba](#) - [Tipo](#) - [Dimensionamiento vertical](#) - [Visibilidad](#) - [Ancho](#)

## Óvalo

Un óvalo estático es un objeto decorativo para los formularios. Los objetos ovalados pueden utilizarse para dibujar formas circulares (cuando las propiedades [ancho](#) y [alto](#) son iguales).



Ejemplo JSON:

```
"myOval": {  
    "type": "oval",           //define el tipo de objeto  
    "left": 60,              //posición izquierda en el formulario  
    "top": 160,              //posición superior en el formulario  
    "width": 100,             //ancho del objeto  
    "height": 20,             //altura del objeto  
    "fill": "blue"           //define el color de fondo  
}
```

## Propiedades soportadas

[Abajo](#) - [Clase](#) - [Tipo de línea punteada](#) - [Color de fondo](#) - [Altura](#) - [Dimensionamiento horizontal](#) - [Izquierda](#) - [Color de línea](#) - [Ancho de línea](#) - [Nombre del objeto](#) - [Derecha](#) - [Arriba](#) - [Tipo](#) - [Dimensionamiento vertical](#) - [Visibilidad](#) - [Ancho](#)

# Spinner

El spinner es un indicador circular que muestra una animación continua, como la [Barber shop](#).



Este tipo de objeto se utiliza para indicar que una operación, como la búsqueda de conexión de red o la realización de un cálculo, está en curso. Cuando se selecciona este indicador, [las propiedades "Graduaciones"](#) no están disponibles.

Cuando se ejecuta el formulario, el objeto no se anima. La animación se gestiona pasando un valor a su [variable o expresión asociada](#):

- 1 (o cualquier valor diferente de 0) = Iniciar la animación,
- 0 = Detener la animación

## Propiedades soportadas

[Estilo de borde](#) - [Abajo](#) - [Clase](#) - [Tipo de expresión](#) - [Altura](#) - [Mensaje de ayuda](#) - [Dim. Horizontal](#) - [Izquierda](#) - [Nombre](#) - [Derecha](#) - [Superior](#) - [Tipo](#) - [Variable o expresión](#) - [Dimensión vertical](#) - [Visibilidad](#) - [Ancho](#)

# Separador

Un separador divide un formulario en dos áreas, permitiendo al usuario ampliar y reducir las áreas moviendo el separador hacia un lado u otro. Un separador puede ser horizontal o vertical. The splitter takes into account each object's resizing properties, which means that you can completely customize your application's interface. Un separador puede ser o no un "empujador."

Los separadores se utilizan, por ejemplo, en los formularios de salida para poder cambiar el tamaño de las columnas:

Job Title:	Company:
Secretary	Howard Battery Co.
Salesperson	Howard Battery Co.
Salesperson	Howard Battery Co.
Supervisor	Howard Battery Co.
Director	BluePines

Algunas de las características generales del separador:

- Puede colocar tantos separadores como desee en todo tipo de formulario y utilizar una mezcla de separadores horizontales y verticales en el mismo formulario.
- Un separador puede cruzar (superponer) un objeto. Este objeto cambiará de tamaño cuando se mueva el separador.
- Los topes de los separadores se calculan para que los objetos desplazados permanezcan totalmente visibles en el formulario o no pasen por debajo/al lado de otro separador. Cuando la propiedad **Empujador** está asociada a un separador, su movimiento hacia la derecha o hacia abajo no encuentra ningún tope.
- Si se redimensiona un formulario mediante un separador, las nuevas dimensiones del formulario se guardan sólo mientras se muestra el formulario. Una vez que se cierra un formulario, se restablecen las dimensiones iniciales.

Una vez insertado, el separador aparece como una línea. Puede modificar su [estilo de borde](#) para obtener una línea más fina o [cambiar su color](#).

Ejemplo JSON:

```
"mySplitter": {  
    "type": "splitter",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20,  
    "splitterMode": "move" //pulsador  
}
```

## Propiedades soportadas

[Border Line Style](#) - [Bottom](#) - [Class](#) - [Height](#) - [Help Tip](#) - [Horizontal Sizing](#) - [Left](#) - [Line Color](#) - [Object Name](#) - [Pusher](#) - [Right](#) - [Top](#) - [Type](#) - [Vertical Sizing](#) - [Variable or Expression](#) - [Visibility](#) - [Width](#)

## Interacción con las propiedades de los objetos vecinos

En un formulario, los separadores interactúan con los objetos que están a su alrededor según las opciones de cambio de tamaño de estos objetos:

Opciones de redimensionamiento de los objetos	Objeto(s) por encima de un separador horizontal o a la izquierda de un separador vertical (1)	Objeto(s) debajo de un separador horizontal <i>no empujador</i> o a la derecha de un separador vertical <i>no empujador</i>	Objeto(s) debajo de un separador horizontal <i>Empujador</i> o a la derecha de un separador vertical <i>Empujador</i>
Ninguno	Permanece como está	Se desplazan con el separador (la posición respecto al separador no se modifica) hasta la siguiente parada. El tope cuando se mueve hacia abajo o hacia la derecha es el borde de la ventana, u otro separador.	Se desplazan con el separador (la posición respecto al separador no se modifica) indefinidamente. No se aplica ninguna parada ( ver el siguiente párrafo)
Redimensionamiento	Conservan la(s) posición(es) original(es), pero se redimensionan según la nueva posición del separador		
Mover	Se mueven con el separador		

(1) *No puede arrastrar el separador más allá del lado derecho (horizontal) o inferior (vertical) de un objeto situado en esta posición.*

Un objeto completamente contenido en el rectángulo que define el separador se mueve al mismo tiempo que el separador.

## Gestión programada de los separadores

Puede asociar un método objeto a un separador y será llamado con el evento `On Clicked` durante todo el movimiento.

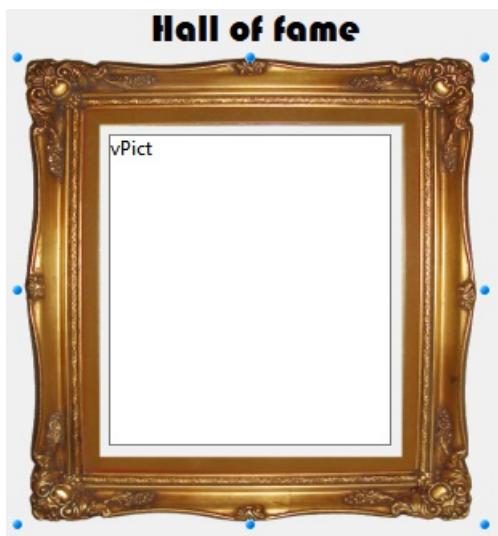
A cada separador se le asocia una [variable](#) de tipo *Longint*. Esta variable se puede utilizar en su objeto y/o métodos de formulario. Su valor indica la posición actual del separador, en píxeles, en relación con su posición inicial.

- Si el valor es negativo: el separador se ha movido hacia arriba o hacia la izquierda,
- Si el valor es positivo: el separador se ha movido hacia el fondo o hacia la derecha,
- Si el valor es 0: el separador se ha movido a su posición original.

También puede mover el separador por programación: sólo tiene que definir el valor de la variable asociada. Por ejemplo, si un separador vertical está asociado a una variable llamada `split1`, y si se ejecuta la siguiente sentencia: `split1:=-10`, el separador se moverá 10 píxeles a la izquierda - como si el usuario lo hiciera manualmente. El movimiento se realiza realmente al final de la ejecución del método del formulario u objeto que contiene la instrucción.

# Imagen estática

Las imágenes estáticas son [objetos estáticos](#) que pueden ser utilizados para varios propósitos en los formularios 4D, incluyendo la decoración, el fondo o la interfaz de usuario:



Las imágenes estáticas se almacenan fuera de los formularios y se insertan por referencia. En el editor de formularios, los objetos imagen estáticos se crean mediante operaciones de copiar/pegar o arrastrar y soltar.

Si coloca una imagen estática en la página 0 de un formulario de varias páginas, aparecerá automáticamente como elemento de fondo en todas las páginas. También puede incluirlo en un formulario heredado, aplicado en el fondo de otros formularios diferentes. Either way, your application will run faster than if the picture was pasted into each page.

## Formato y ubicación

La imagen original debe estar almacenada en un formato gestionado de forma nativa por 4D (4D reconoce los principales formatos de imagen: JPEG, PNG, BMP, SVG, GIF, etc.).

Se pueden utilizar dos ubicaciones principales para la trayectoria de la imagen estática:

- in the Resources folder of the project. Appropriate when you want to share static pictures between several forms in the project. En este caso, la ruta de acceso está en "/RESOURCES/<picture path>".
- en una carpeta de imágenes (por ejemplo, llamada `Images`) dentro de la carpeta del formulario. Conveniente cuando las imágenes estáticas se utilizan sólo en el formulario y/o se quiere poder mover o duplicar todo el formulario dentro del proyecto o de diferentes proyectos. En este caso, el nombre de la ruta es "<¥picture path>" y se resuelve desde la raíz de la carpeta del formulario.

## Propiedades soportadas

[Abajo](#) - [Clase CSS Class](#) - [Mostrar](#) - [Altura](#) - [Dim. horizontal](#) - [Izquierda](#) - [Nombre del objeto](#) - [Nombre de ruta](#) - [Derecha](#) - [Arriba](#) - [Tipo](#) - [Dim. vertical](#) - [Visibilidad](#) - [Ancho](#)

# Stepper

Un stepper permite al usuario desplazarse por valores numéricos, duraciones (tiempos) o fechas por pasos predefinidos haciendo clic en los botones de flecha.

Stepper associated with `vStep` variable



## Uso del stepper

Puede asignar la variable asociada al objeto a un área introducible (campo o variable) para almacenar o modificar el valor actual del objeto.

Se puede asociar un stepper directamente a una variable numérica, hora o fecha.

- Para los valores de tipo hora, las propiedades Mínimo, Máximo y Paso representan segundos. Por ejemplo, para programar un paso de 8:00 a 18:00 con pasos de 10 minutos:
  - `minimum` = 28 800 (8\*60\*60)
  - `maximum` = 64 800 (18\*60\*60)
  - `step` = 600 (10\*60)
- Para los valores de tipo fecha, el valor introducido en la propiedad `paso` representa días. Las propiedades Mínimo y Máximo se ignoran.

Para que el stepper trabaje con una variable de hora o fecha, es imprescindible definir su tipo en la Lista de propiedades Y declararlo explícitamente a través del comando `C_TIME` o `C_DATE`.

For more information, please refer to [Using indicators](#) in the "Progress Indicator" page.

## Propiedades soportadas

[Border Line Style](#) - [Bottom](#) - [Class](#) - [Enterable](#) - [Execute object method](#) - [Expression Type](#) (only "integer", "number", "date", or "time") - [Height](#) - [Help Tip](#) - [Horizontal Sizing](#) - [Left](#) - [Maximum](#) - [Minimum](#) - [Object Name](#) - [Right](#) - [Step](#) - [Top](#) - [Type](#) - [Variable or Expression](#) - [Vertical Sizing](#) - [Visibility](#) - [Width](#)

## Ver también

- [indicadores de progreso](#)
- [reglas](#)

# Subformulario

Un subformulario es un formulario incluido en otro formulario.

## Terminología

Con el fin de definir claramente los conceptos implementados con los subformularios, aquí hay algunas definiciones para ciertos términos utilizados:

- Subformulario: un formulario destinado a ser incluido en otro formulario, llamado a su vez formulario padre.
- Formulario padre: un formulario que contiene uno o más subformularios.
- Contenedor de subformulario: un objeto incluido en el formulario padre, que muestra una instancia del subformulario.
- Instancia de subformulario: la representación de un subformulario en un formulario padre. Este concepto es importante porque es posible mostrar varias instancias del mismo subformulario en un formulario padre.
- Formulario listado: instancia de subformulario mostrada como una lista.
- Formulario detallado: formulario de entrada tipo página asociado a un subformulario tipo lista al que se accede haciendo doble clic en la lista.

## Sub-formularios en lista

Un subformulario lista le permite introducir, ver y modificar datos en otras tablas. Normalmente se utilizan subformularios lista en bases de datos en las que se han establecido relaciones Uno a Muchos. Un subformulario lista en un formulario de una tabla Uno relacionada le permite ver, introducir y modificar datos en una tabla Muchos relacionada. Puede tener varios subformularios procedentes de diferentes tablas en el mismo formulario. Sin embargo, no es posible colocar dos subformularios que pertenecen a la misma tabla en la misma página de un formulario.

Por ejemplo, una base de datos del gestor de contactos puede utilizar un subformulario lista para mostrar todos los números de teléfono de un contacto. Aunque los números de teléfono aparecen en la pantalla Contactos, la información se almacena realmente en una tabla relacionada. Utilizando una relación de Uno a Muchos, este diseño de base de datos facilita el almacenamiento de un número ilimitado de números de teléfono por contacto. Con las relaciones automáticas, se puede soportar la entrada de datos directamente en la tabla Muchos relacionada sin programar.

Aunque los subformularios lista suelen estar asociados a muchas tablas, una instancia de subformulario puede mostrar los registros de cualquier otra tabla de la base de datos.

También puede permitir que el usuario introduzca datos en el formulario lista. Dependiendo de la configuración del subformulario, el usuario puede mostrar el formulario detallado haciendo doble clic en un subregistro o utilizando los comandos para añadir y editar subregistros.

4D ofrece tres acciones estándar para satisfacer las necesidades básicas de gestión de los subregistros: [Edit](#), [Subrecord](#), [Delete Subrecord](#) y [Add Subrecord](#). Cuando el formulario incluye varias instancias de subformulario, la acción se aplicará al subformulario que tenga el foco.

## Sub-formularios en página

Los subformularios en modo página pueden mostrar los datos del subregistro actual o todo valor pertinente en función del contexto (variables, imágenes, etc.). Una de las principales ventajas de utilizar subformularios página es que pueden incluir funcionalidades avanzadas y pueden interactuar directamente con el formulario padre (widgets). Los subformularios en página también tienen sus propiedades y eventos específicos; puede gestionarlos completamente por programación.

El subformulario en página utiliza el formulario de entrada indicado por la propiedad [Formulario detallado](#). A diferencia de un subformulario en modo lista, el formulario utilizado puede proceder de la misma tabla que el formulario padre. También es posible utilizar un formulario proyecto. Cuando se ejecuta, un subformulario en modo página tiene las

mismas características de visualización estándar que un formulario de entrada.

Los widgets 4D son objetos compuestos predefinidos basados en subformularios página. Se describen detalladamente en un manual aparte, [4D Widgets](#).

## Using the bound variable or expression

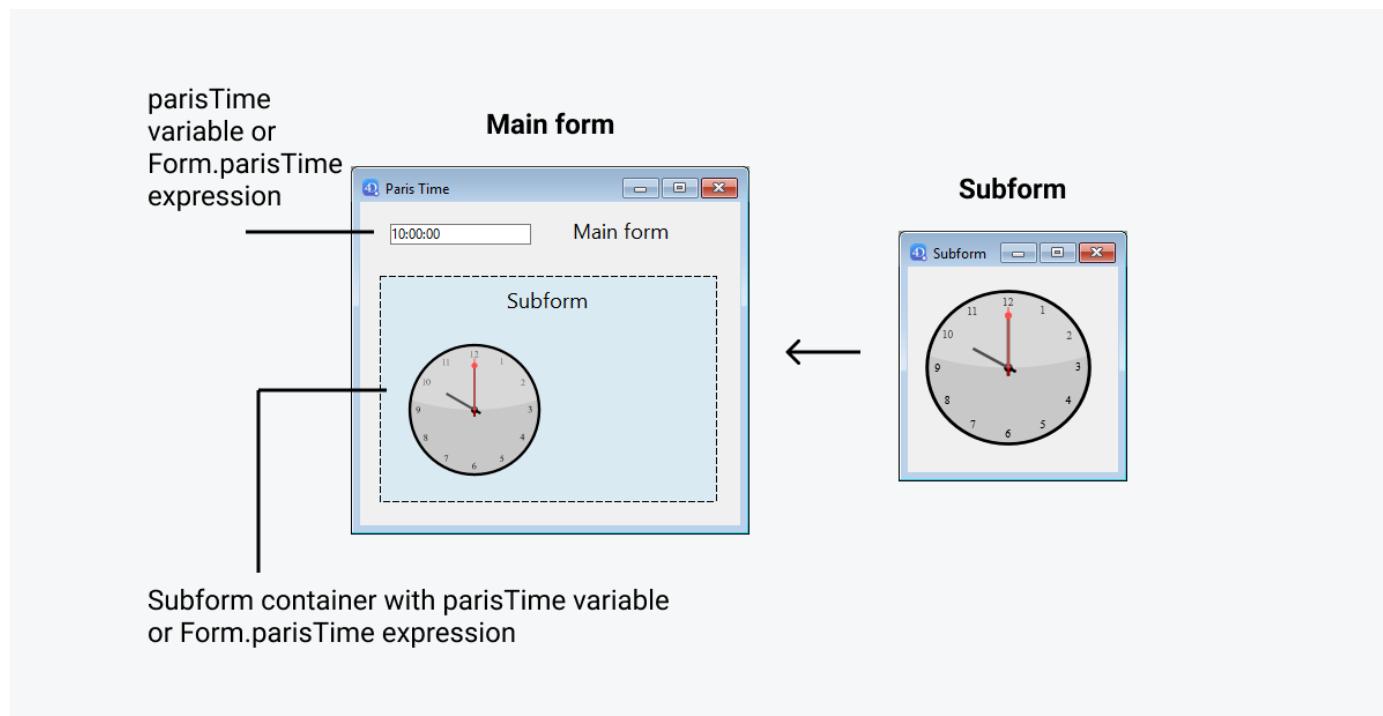
You can bind [a variable or an expression](#) to a subform container object. This is very useful to synchronize values from the parent form and its subform(s).

By default, 4D creates a variable or expression of [object type](#) for a subform container, which allows you to share values in the context of the subform using the [Form](#) command ([see below](#)). However, you can use a variable or expression of any scalar type (time, integer, etc.) especially if you only need to share a single value:

- Define a bound variable or expression of a scalar type and call the [OBJECT Get subform container value](#) and [OBJECT SET SUBFORM CONTAINER VALUE](#) commands to exchange values when [On Bound Variable Change](#) or [On Data Change](#) form events occur. This solution is recommended to synchronize a single value.
- Define a bound variable or expression of the object type and use the [Form](#) command to access its properties from the subform. This solution is recommended to synchronize several values.

## Synchronizing parent form and subform (single value)

Binding the same variable or expression to your subform container and other objects of the parent form lets you link the parent form and subform contexts to put the finishing touches on sophisticated interfaces. Imagine a subform representing a clock, inserted into a parent form containing an enterable variable of the Time type:



In the parent form, both objects (time variable and subform container) \*\*\*have the same value as \*\*\*Variable or Expression\*\*\*\*\*. It can be a variable (e.g. `parisTime`), or an expression (e.g. `Form.parisTime`).

In the subform, the clock object is managed through the `Form.clockValue` property.

### Updating the contents of a subform

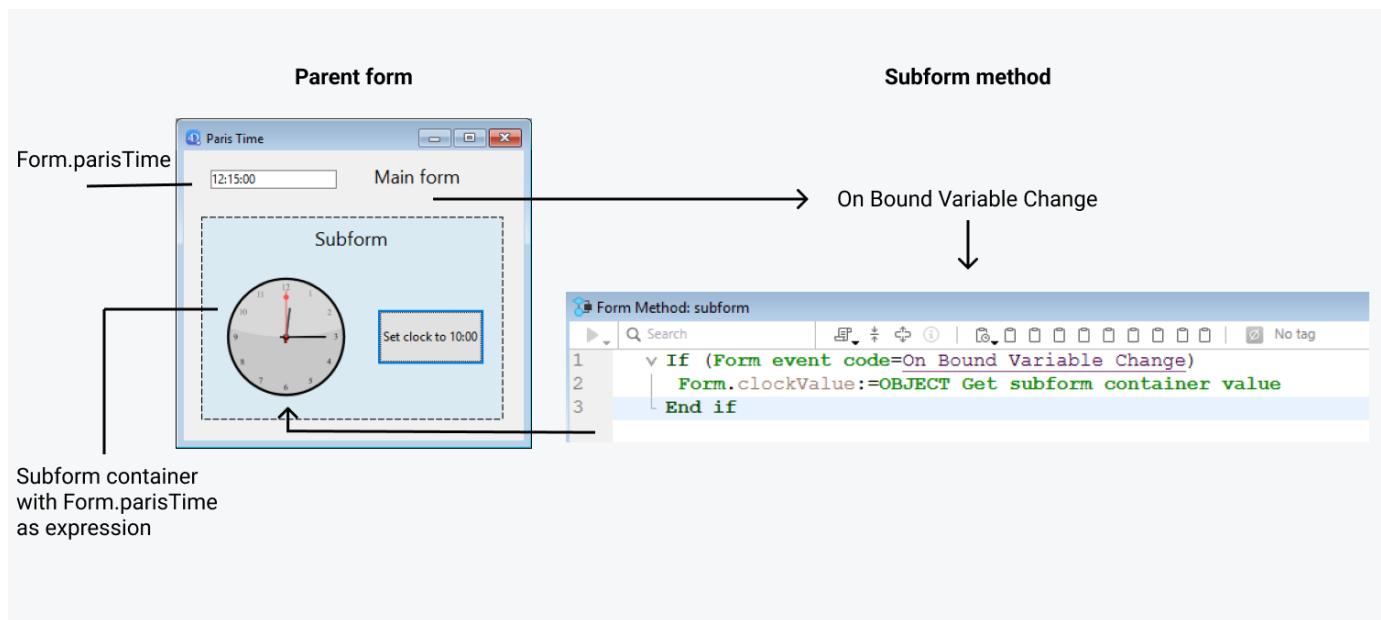
Case 1: The value of the parent form variable or expression is modified and this modification must be passed on to a subform.

`Form.parisTime` changes to 12:15:00 in the parent form, either because the user entered it, or because it was updated dynamically (via the `Current time` command for example). This triggers the [On Bound Variable Change](#) event in the subform's Form method.

Se ejecuta el siguiente código:

```
// Subform form method
If (Form event code=On Bound Variable Change) //bound variable or expression was modified in the parent
    Form.clockValue:=OBJECT Get subform container value //synchronize the local value
End if
```

It updates the value of `Form.clockValue` in the subform:



The [On Bound Variable Change](#) form event is generated:

- as soon as a value is assigned to the variable/expression of the parent form, even if the same value is reassigned
- si el subformulario pertenece a la página formulario actual o a la página 0.

Note that, as in the above example, it is preferable to use the `OBJECT Get subform container value` command which returns the value of the expression in the subform container rather than the expression itself because it is possible to insert several subforms in the same parent form (for example, a window displaying different time zones contains several clocks).

Modifying the bound variable or expression triggers form events which let you synchronize the parent form and subform values:

- Use the [On Bound Variable Change](#) form event to indicate to the subform (form method of subform) that the variable or expression was modified in the parent form.
- Use the [On Data Change](#) form event to indicate to the subform container that the variable or expression value was modified in the subform.

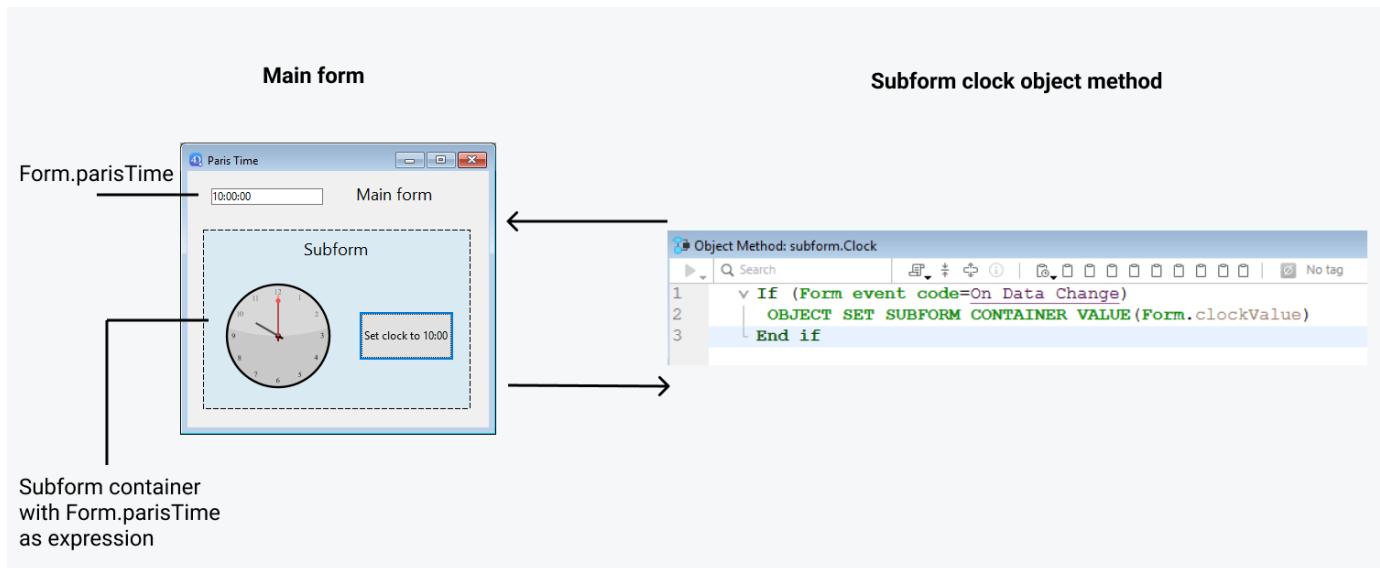
#### Actualizar el contenido de un formulario padre

Caso 2: se modifica el contenido del subformulario y esta modificación debe pasar al formulario padre.

Inside the subform, the button changes the value of the `Form.clockValue` expression of type Time attached to the clock object. This triggers the [On Data Change](#) form event inside the clock object (this event must be selected for the object), which updates the `Form.parisTime` value in the main form.

Se ejecuta el siguiente código:

```
// subform clock object method
If (Form event code=On Data Change) //whatever the way the value is changed
    OBJECT SET SUBFORM CONTAINER VALUE(Form.clockValue) //Push the value to the container
End if
```



Everytime the value of `Form.clockValue` changes in the subform, `Form.parisTime` in the subform container is also updated.

If the variable or expression value is set at several locations, 4D uses the value that was loaded last. It applies the following loading order: 1-Object methods of subform, 2-Form method of subform, 3-Object methods of parent form, 4-Form method of parent form

## Synchronizing parent form and subform (multiple values)

By default, 4D binds a variable or expression of [object type](#) to each subform. The contents of this object can be read and/or modified from within the parent form and from the subform, allowing you to share multiple values in a local context.

When bound to the subform container, this object is returned by the `Form` command directly in the subform. Since objects are always passed by reference, if the user modifies a property value in the subform, it will automatically be saved in the object itself and thus, available to the parent form. On the other hand, if a property of the object is modified by the user in the parent form or by programming, it will be automatically updated in the subform. No event management is necessary.

For example, in a subform, inputs are bound to the `Form` object properties (of the subform form):

Lastname:	<code>Form.lastname</code>
Firstname:	<code>Form.firstname</code>

En el formulario padre, se muestra el subformulario dos veces. Each subform container is bound to an expression which is a property of the `Form` object (of the parent form):

<b>Father</b>	<code>Form.father.lastname</code>	Form or Expression: <b>Form.father</b>
Lastname:	<code>Form.lastname</code>	
Firstname:	<code>Form.firstname</code>	
<b>Mother</b>	<code>Form.mother.lastname</code>	Form or Expression: <b>Form.mother</b>
Lastname:	<code>Form.lastname</code>	
Firstname:	<code>Form.firstname</code>	
<b>Add values</b>		

The button only creates `mother` and `father` properties in the parent's `Form` object:

```
//Add values button object method
Form.mother:=New object("lastname"; "Hotel"; "firstname"; "Anne")
Form.father:=New object("lastname"; "Golf"; "firstname"; "Félix")
```

When you execute the form and click on the button, you see that all values are correctly displayed:

The screenshot shows a 4D form with two subforms. The top subform is titled 'Father' and contains fields for Lastname ('Golf') and Fristname ('Félix'). The bottom subform is titled 'Mother' and contains fields for Lastname ('Hotel') and Fristname ('Anne'). At the bottom of the main form is a button labeled 'Add values'.

If you modify a value either in the parent form or in the subform, it is automatically updated in the other form because the same object is used:

The screenshot shows the same 4D form after modifications. The 'Father' subform's Lastname field now contains 'Wolf'. The 'Mother' subform's Lastname field now contains 'Hotelle'. Red arrows point from the modified values ('Wolf' and 'Hotelle') back to their respective original values ('Golf' and 'Hotel') in the other form.

## Using pointers (compatibility)

In versions prior to 4D v19 R5, synchronization between parent forms and subforms was handled through pointers. For example, to update a subform object, you could call the following code:

```
// Subform form method
If (Form event code=On Bound Variable Change)
    ptr:=OBJECT Get pointer(Object subform container)
    clockValue:=ptr->
End if
```

This principle is still supported for compatibility but is now deprecated since it does not allow binding expressions to subforms. It should no longer be used in your developments. In any cases, we recommend to use the `Form` command or the `OBJECT Get subform container value` and `OBJECT SET SUBFORM CONTAINER VALUE` commands to synchronize form and subform values.

## Programación entre formularios avanzada

Communication between the parent form and the instances of the subform may require going beyond the exchange of a values through the bound variable. De hecho, es posible que desee actualizar las variables de los subformularios en

función de las acciones realizadas en el formulario principal y viceversa. Si utilizamos el ejemplo anterior del subformulario de tipo "reloj dinámico", es posible que queramos definir una o varias horas de alarma para cada reloj.

4D ha implementado los siguientes mecanismos para satisfacer estas necesidades:

- Calling of a container object from the subform using the `CALL SUBFORM CONTAINER` command
- Execution of a method in the context of the subform via the `EXECUTE METHOD IN SUBFORM` command

El comando `GOTO OBJECT` busca el objeto de destino en el formulario padre aunque se ejecute desde un subformulario.

### Comando CALL SUBFORM CONTAINER

The `CALL SUBFORM CONTAINER` command lets a subform instance send an `event` to the subform container object, which can then process it in the context of the parent form. El evento se recibe en el método del objeto contenedor. Puede estar en el origen de todo evento detectado por el subformulario (clic, arrastrar y soltar, etc.).

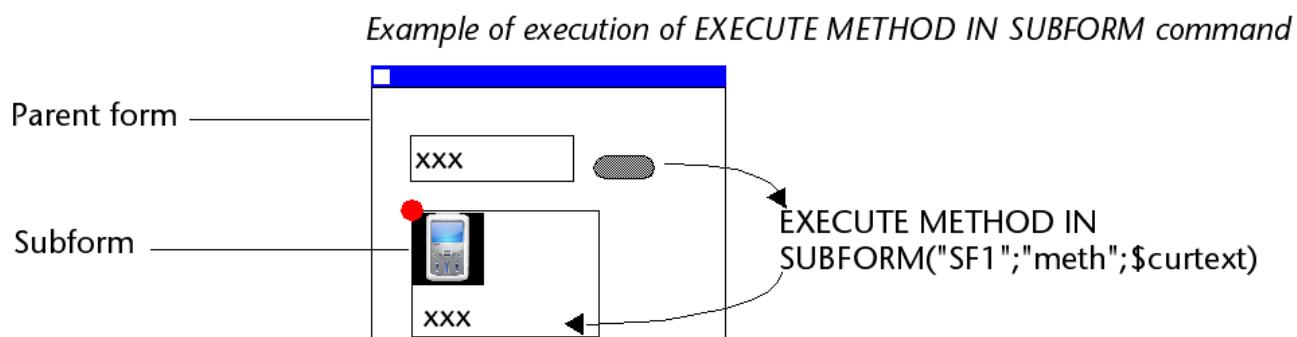
El código del evento no tiene restricciones (por ejemplo, 20000 o -100). Puede utilizar un código que corresponda a un evento existente (por ejemplo, 3 para `On Validate`), o utilizar un código personalizado. En el primer caso, sólo puede utilizar los eventos que haya marcado en la lista de propiedades para los contenedores de subformulario. En el segundo caso, el código no debe corresponder a ningún evento de formulario existente. Se recomienda utilizar un valor negativo para asegurarse de que este código no será utilizado por 4D en futuras versiones.

Para más información, consulte la descripción del comando `CALL SUBFORM CONTAINER`.

### Comando EXECUTE METHOD IN SUBFORM

El comando `EXECUTE METHOD IN SUBFORM` permite que un formulario o uno de sus objetos solicite la ejecución de un método en el contexto de la instancia del subformulario, lo que le da acceso a las variables, objetos, etc. del subformulario. Este método también puede recibir parámetros.

Este mecanismo se ilustra en el siguiente diagrama:



Para más información, consulte la descripción del comando `EXECUTE METHOD IN SUBFORM`.

### Propiedades soportadas

Border Line Style - Bottom - Class - Detail Form - Double click on empty row - Double click on row - Enterable in list - Expression Type - Focusable - Height - Hide focus rectangle - Horizontal Scroll Bar - Horizontal Sizing - Left - List Form - Method - Object Name - Print Frame - Right - Selection mode - Source - Top - Type - Variable or Expression - Vertical Scroll Bar - Vertical Sizing - Visibility - Width

# Pestañas

Un control de pestañas crea un objeto que permite al usuario elegir entre un conjunto de pantallas virtuales que están encerradas por el objeto de control de pestañas. Se accede a cada pantalla haciendo clic en su pestaña.

El siguiente formulario multipágina utiliza un objeto de control de pestañas:

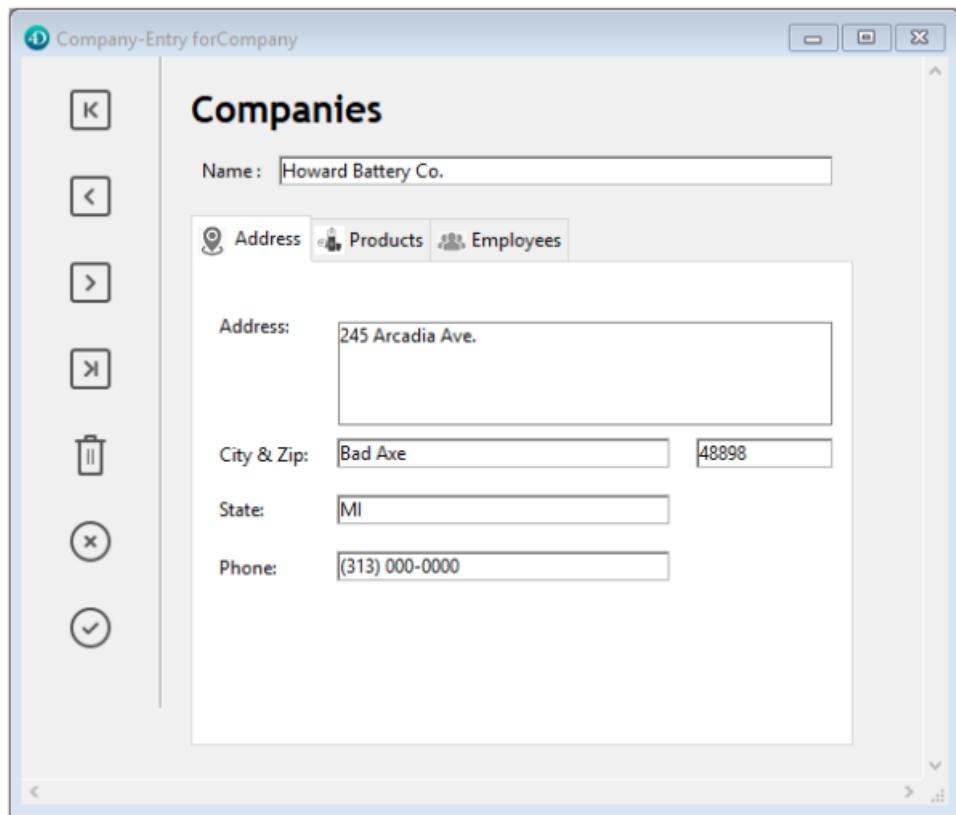
The screenshot shows a Windows-style application window titled "Company-Entry". On the left side, there is a vertical toolbar with icons for search (magnifying glass), back (left arrow), forward (right arrow), delete (trash can), and checkmark (checkmark). The main area is titled "Companies". It contains a text input field labeled "Name" with the value "Howard Battery Co.". Below this is a tab control with three tabs: "Address", "Products", and "Employees". The "Address" tab is selected and active. Inside the "Address" tab, there are four input fields: "Address" (containing "245 Arcadia Ave."), "City & Zip" (containing "Bad Axe" and "48898"), "State" (containing "MI"), and "Phone" (containing "(313) 000-0000").

Para navegar de una pantalla a otra, el usuario sólo tiene que hacer clic en la pestaña deseada.

Las pantallas pueden representar páginas en un formulario de varias páginas o un objeto que cambia cuando el usuario hace clic en una pestaña. If the tab control is used as a page navigation tool, then the `FORM GOTO PAGE` command or the `gotoPage` standard action would be used when a user clicks a tab.

Otro uso del control de pestañas es para controlar los datos que se muestran en un subformulario. Por ejemplo, se podría implementar un Rolodex utilizando un control de pestañas. Las pestañas mostrarían las letras del alfabeto y la acción del control de pestañas sería cargar los datos correspondientes a la letra que el usuario pulsara.

Cada pestaña puede mostrar etiquetas o rótulos y un pequeño ícono. Si incluye iconos, éstos aparecen a la izquierda de cada etiqueta. Este es un ejemplo de un control de pestañas que utiliza iconos:



Cuando se crea un control de pestañas, 4D gestiona el espacio y la colocación de las mismas. Sólo tiene que suministrar las etiquetas en forma de array, o los iconos y etiquetas en forma de lista jerárquica.

Si el control de pestañas es lo suficientemente amplio como para mostrar todas las pestañas con las etiquetas y los iconos, muestra ambos. Si el control de pestañas no es lo suficientemente ancho para mostrar tanto las etiquetas como los iconos, 4D muestra sólo los iconos. Si no caben todos los iconos, coloca flechas de desplazamiento a la derecha de la última pestaña visible. Las flechas de desplazamiento permiten al usuario desplazar los iconos hacia la izquierda o la derecha.

En macOS, además de la posición estándar (arriba), los controles de las pestañas también pueden alinearse en la parte inferior.

## Ejemplo JSON:

```

"myTab": {
    "type": "tab",
    "left": 60,
    "top": 160,
    "width": 100,
    "height": 20,
    "labelsPlacement": "bottom" //define la orientación
}

```

## Añadir etiquetas a un control de pestañas

Para suministrar las etiquetas de un control de pestañas, puede utilizar:

- un objeto
- una lista de selección
- un array

### Utilizar un objeto

You can assign an **object** encapsulating a **collection** as the **data source** of the tab control. El objeto debe contener las siguientes propiedades:

Propiedad	Tipo	Descripción
values	Collection	Obligatorio - Colección de valores escalares. Sólo se admiten valores de tipo cadena. If invalid, empty or not defined, the tab control is empty
index	number	Index of the currently tab control page (value between 0 and <code>collection.length-1</code> )
currentValue	Texto	Valor actual seleccionado

The initialization code must be executed before the form is presented to the user.

In the following example, `Form.tabControl` has been defined as tab control [expression](#). You can associate the [gotoPage standard action](#) to the form object:

```
Form.tabControl:=New object
Form.tabControl.values:=New collection("Page 1"; "Page 2"; "Page 3")
Form.tabControl.index:=2 //start on page 3
```

## Utilizar una lista de selección

You can assign a [choice list](#) to the tab control, either through a collection (static list) or a JSON pointer to a json list ("\$ref"). Icons associated with list items in the Lists editor will be displayed in the tab control.

## Utilizar un array Text

Puede crear un array Texto que contenga los nombres de cada página del formulario. Este código debe ejecutarse antes de que el formulario se presente al usuario. Por ejemplo, podrías colocar el código en el método del objeto del control de la pestaña y ejecutarlo cuando se produzca el evento `On Load`.

```
ARRAY TEXT(arrPages;3)
arrPages{1}:="Name"
arrPages{2}:="Address"
arrPages{3}:="Notes"
```

You can also store the names of the pages in a hierarchical list and use the [LIST TO ARRAY](#) command to load the values into the array.

## Funcionalidades de Goto page

### Comando FORM GOTO PAGE

You can use the `FORM GOTO PAGE` command in the tab control's method:

```
FORM GOTO PAGE(arrPages)
```

The command is executed when the `On Clicked` event occurs. You should then clear the array when the `On Unload` event occurs.

He aquí un ejemplo de método objeto:

```
Case of
:(Form event=On Load)
  LIST TO ARRAY("Tab Labels";arrPages)
:(Form event=On Clicked)
  FORM GOTO PAGE(arrPages)
:(Form event=On Unload)
  CLEAR VARIABLE(arrPages)
End case
```

## Acción Goto Page

Cuando se asigna la acción [acción estándar](#) `gotoPage` a un control de pestañas, 4D mostrará automáticamente la página del formulario que corresponde al número de la pestaña que está seleccionada.

Por ejemplo, si el usuario selecciona la tercera pestaña, 4D mostrará la página 3 del formulario actual (si existe).

## Propiedades soportadas

[Bold](#) - [Bottom](#) - [Choice List](#) - [Class](#) - [Expression Type](#) - [Font](#) - [Font Size](#) - [Height](#) - [Help Tip](#) - [Horizontal Sizing](#) - [Italic](#) - [Left](#) - [Object Name](#) - [Right](#) - [Save value](#) - [Standard action](#) - [Tab Control Direction](#) - [Top](#) - [Type](#) - [Underline](#) - [Vertical Sizing](#) - [Variable or Expression](#) - [Visibility](#) - [Width](#)

# Texto

Un objeto texto permite mostrar contenido escrito estático (\*por ejemplo, \*, instrucciones, títulos, etiquetas, etc.) en un formulario. Estas áreas de texto estáticas pueden convertirse en dinámicas cuando incluyen referencias dinámicas. Para más información, consulte [Uso de referencias en textos estáticos](#).

Ejemplo JSON:

```
"myText": {  
    "type": "text",  
    "text": "Hello World!",  
    "textAlign": "center",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20,  
    "stroke": "#ff0000"      //color texto  
    "fontWeight": "bold"  
}
```

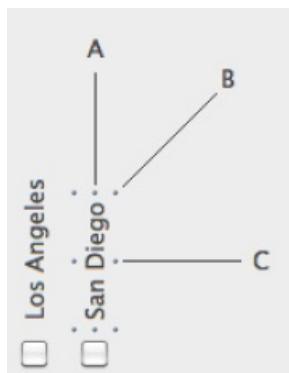
## Rotación

4D le permite rotar las áreas de texto en sus formularios utilizando la propiedad [Orientación](#).

	New York	Chicago	Los Angeles	San Diego
Alpha	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Bravo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Charlie	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Delta	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Echo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

La rotación del texto puede definirse para un proceso utilizando el comando de lenguaje `OBJECT SET TEXT ORIENTATION`.

Una vez que un texto está rotado, puede seguir cambiando su tamaño o posición, así como todas sus propiedades. Tenga en cuenta que las propiedades de alto y ancho del área de texto no dependen de su orientación:



- Si el objeto se redimensiona en la dirección A, se modifica su [ancho](#);
- Si el objeto se redimensiona en la dirección C, se modifica su [alto](#);

- Si el objeto se redimensiona en la dirección B, se modifican tanto su [ancho](#) como su [alto](#).

## Propiedades soportadas

Negrita - Estilo del borde - Abajo - Clase - Color de relleno - Fuente - Color de la fuente - Tamaño de fuente - Altura - Alineación horizontal - Dimensionamiento horizontal - Itálica - Izquierda - Nombre del objeto - Orientación - Derecha - Título - Arriba - Tipo - Subrayado - Dimensionamiento vertical - Visibilidad - Ancho

# Área Web

Las áreas web pueden mostrar varios tipos de contenido web dentro de sus formularios: páginas HTML con contenidos estáticos o dinámicos, archivos, imágenes, JavaScript, etc. El motor de renderizado del área web depende de la plataforma de ejecución de la aplicación y de la opción motor de renderizado seleccionada.

Es posible crear varias áreas web en el mismo formulario. Tenga en cuenta, sin embargo, que el uso de las áreas web debe seguir [varias reglas](#).

Varias [acciones estándar](#) dedicadas, numerosos [comandos de lenguaje](#) así como también [eventos formulario](#) genéricos y específicos, permiten al desarrollador controlar el funcionamiento de las áreas web. Se pueden utilizar variables específicas para intercambiar información entre el área y el entorno 4D.

## Propiedades específicas

### Variables asociadas

Se pueden asociar dos variables específicas a cada área web:

- [URL](#) --para controlar la URL que muestra el área web
- [Progresión](#) -- para controlar el porcentaje de carga de la página mostrada en el área web.

A partir de 4D v19 R5, la variable Progression ya no se actualiza en las Áreas Web que utilizan el [motor de renderizado del sistema Windows](#).

### Motor de renderización web

Puede elegir entre [dos motores de renderizado](#) para el área web, dependiendo de las particularidades de su aplicación.

Seleccionar el motor de renderizado web anidado permite llamar a los métodos de 4D desde el área web y asegurarse de que las funcionalidades en macOS y Windows sean similares. Se recomienda seleccionar el motor de renderizado del sistema cuando el área web está conectada a Internet porque siempre se beneficia de las últimas actualizaciones de seguridad.

### Acceder a los métodos 4D

Cuando se selecciona la propiedad [Acceder a los métodos 4D](#), se puede llamar a los métodos 4D desde un área web.

Esta propiedad sólo está disponible si el área web [utiliza el motor de renderizado web integrado](#).

### Objeto \$4d

El [motor de renderizado web integrado 4D](#) suministra al área un objeto JavaScript llamado \$4d que puede asociar a cualquier método proyecto 4D utilizando la notación objeto ".":

Por ejemplo, para llamar al método [HelloWorld](#) de 4D, basta con ejecutar la siguiente declaración:

```
$4d.HelloWorld();
```

JavaScript es sensible a las mayúsculas y minúsculas, por lo que es importante tener en cuenta que el objeto se llama \$4d (con "d" minúscula).

La sintaxis de las llamadas a los métodos 4D es la siguiente:

```
$4d.4DMethodName(param1,paramN,function(result){})
```

- `param1...paramN` : Puede pasar tantos parámetros como necesite al método 4D. Estos parámetros pueden ser de cualquier tipo soportado por JavaScript (cadena, número, array, objeto).
- `function(result)` : Función a pasar como último argumento. Esta función "callback" se llama de forma sincrónica una vez que el método 4D termina de ejecutarse. Recibe el parámetro `result`.
- `result` : resultado de la ejecución del método 4D, devuelto en la expresión "\$0". Este resultado puede ser de cualquier tipo soportado por JavaScript (cadena, número, array, objeto). Puede utilizar el comando `C_OBJECT` para devolver los objetos.

Por defecto, 4D trabaja en UTF-8. Cuando devuelva un texto que contenga caracteres extendidos, por ejemplo, caracteres con acentos, asegúrese de que la codificación de la página mostrada en el área web esté declarada como UTF-8, ya que de lo contrario los caracteres podrían representarse incorrectamente. En este caso, añada la siguiente línea en la página HTML para declarar la codificación: `<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />`

## Ejemplo 1

Dado un método proyecto 4D llamado `today` que no recibe parámetros y devuelve la fecha actual como una cadena.

Código 4D del método `today`:

```
C_TEXT($0)
$0:=String(Current date;System date long)
```

En el área web, el método 4D puede ser llamado con la siguiente sintaxis:

```
$4d.today()
```

El método 4D no recibe ningún parámetro pero devuelve el valor \$0 a la función callback llamada por 4D tras la ejecución del método. Queremos mostrar la fecha en la página HTML que es cargada por el área web.

Aquí está el código de la página HTML:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<script type="text/javascript">
$4d.today(function(dollarZero)
{
    var curDate = dollarZero;
    document.getElementById("mydiv").innerHTML=curDate;
});
</script>
</head>
<body>Today is: <div id="mydiv"></div>
</body>
</html>
```

## Ejemplo 2

El método proyecto 4D `calcSum` recibe los parámetros ( `$1...$n` ) y devuelve su suma en `$0`:

Código 4D del método `calcSum`:

```

C_REAL(${1}) // recibe n parámetros de tipo REAL
C_REAL($0) // devuelve un Real
C_LONGINT($i;$n)
$n:=Count parameters
For($i;1;$n)
    $0:=$0+${$i}
End for

```

El código JavaScript que se ejecuta en el área web es el siguiente:

```

$4d.calcSum(33, 45, 75, 102.5, 7, function(dollarZero)
{
    var result = dollarZero // el resultado es 262.5
});

```

## Acciones estándar

Hay cuatro acciones estándar específicas para gestionar las áreas web de forma automática: `Open Back URL`, `Open Forward URL`, `Refresh Current URL` y `Stop Loading URL`. Estas acciones pueden asociarse a botones o comandos de menú y permiten una rápida implementación de interfaces web básicas. Estas acciones se describen en [Acciones estándar](#).

## Eventos formulario

Los eventos formulario específicos están destinados a la gestión programada de las áreas de la web, más concretamente a la activación de los enlaces:

- [On Begin URL Loading](#)
- [On URL Resource Loading](#)
- [On End URL Loading](#)
- [On URL Loading Error](#)
- [On URL Filtering](#)
- [On Open External Link](#)
- [On Window Opening Denied](#)

Además, las áreas web soportan los siguientes eventos de formulario genéricos:

- [On Load](#)
- [On Unload](#)
- [On Getting Focus](#)
- [On Losing Focus](#)

## Reglas de las áreas web

### Interfaz de usuario

Cuando se ejecuta el formulario, las funciones estándar de la interfaz del navegador están disponibles para el usuario en el área web, lo que permite la interacción con otras áreas del formulario:

- Comandos menú Edición: cuando el área web tiene el foco, los comandos del menú Edición pueden utilizarse para realizar acciones como copiar, pegar, seleccionar todo, etc., según la selección.
- El menú contextual: es posible utilizar el [menú contextual](#) estándar del sistema con el área web. La visualización del menú contextual se puede controlar con el comando `WA SET PREFERENCE`.
- Arrastrar y soltar: el usuario puede arrastrar y soltar texto, imágenes y documentos dentro del área web o entre un área web y los objetos de los formularios 4D, según las propiedades de los objetos 4D. Por razones de seguridad,

no se permite por defecto cambiar el contenido de un área web mediante la acción de arrastrar y soltar un archivo o una URL. En este caso, el cursor muestra un ícono "prohibido"  . Tiene que utilizar la instrucción `WA SET PREFERENCE(*; "warea";WA enable URL drop;True)` para mostrar un ícono "soltar" y generar el evento `On Window Opening Denied`. En este evento, se puede llamar al comando `WA OPEN URL` o establecer la variable `URL` en respuesta a una caída del usuario.

Las funciones de arrastrar y soltar descritas anteriormente no son compatibles con las áreas web que utilizan el motor de renderización del sistema [de macOS](#).

## Subformularios

Por razones relacionadas con los mecanismos de redibujado de ventanas, la inserción de un área web en un subformulario está sujeta a las siguientes restricciones:

- El subformulario no debe poder desplazarse
- Los límites del área web no deben superar el tamaño del subformulario

No se soporta la superposición de un área web sobre o debajo de otros objetos formulario.

## Conflicto entre el área web y el servidor web (Windows)

En Windows, no se recomienda acceder, a través de un área web, al servidor web de la aplicación 4D que contiene el área, ya que esta configuración podría provocar un conflicto que paralice la aplicación. Por supuesto, un 4D remoto puede acceder al servidor web de 4D Server, pero no a su propio servidor web.

## Inserción del protocolo (macOS)

Las URLs manejadas por programación en áreas web bajo macOS deben comenzar con el protocolo. Por ejemplo, debe pasar la cadena "<http://www.mysite.com>" y no sólo "[www.mysite.com](http://www.mysite.com)".

## Acceso al inspector web

Puede visualizar y utilizar un inspector web dentro de las áreas web de sus formularios o en las áreas web fuera de la pantalla. El inspector web es un depurador que permite analizar el código y el flujo de información de las páginas web.

Para mostrar el inspector Web, puede ejecutar el comando `WA OPEN WEB INSPECTOR` o utilizar el menú contextual del área web.

- \*\*Ejecute el comando `WA OPEN WEB INSPECTOR` \*\*  
Este comando se puede utilizar directamente con áreas web en pantalla (objeto formulario) y fuera de ella.
- Use the web area context menu  
This feature can only be used with onscreen web areas and requires that the following conditions are met:
  - el [menú contextual](#) del área web está activado
  - el uso del inspector está expresamente autorizado en el área mediante la siguiente declaración:

```
WA SET PREFERENCE(*;"WA";WA enable Web inspector;True)
```

Con [Motor de renderizado del sistema Windows](#), un cambio en esta preferencia requiere que se tenga en cuenta una acción de navegación en el área (por ejemplo, una actualización de la página).

Para más información, consulte la descripción del comando `WA SET PREFERENCE`.

Cuando haya realizado los ajustes como se ha descrito anteriormente, entonces tendrá nuevas opciones como Inspeccionar elemento en el menú contextual del área. Al seleccionar esta opción, se muestra la ventana del inspector web.

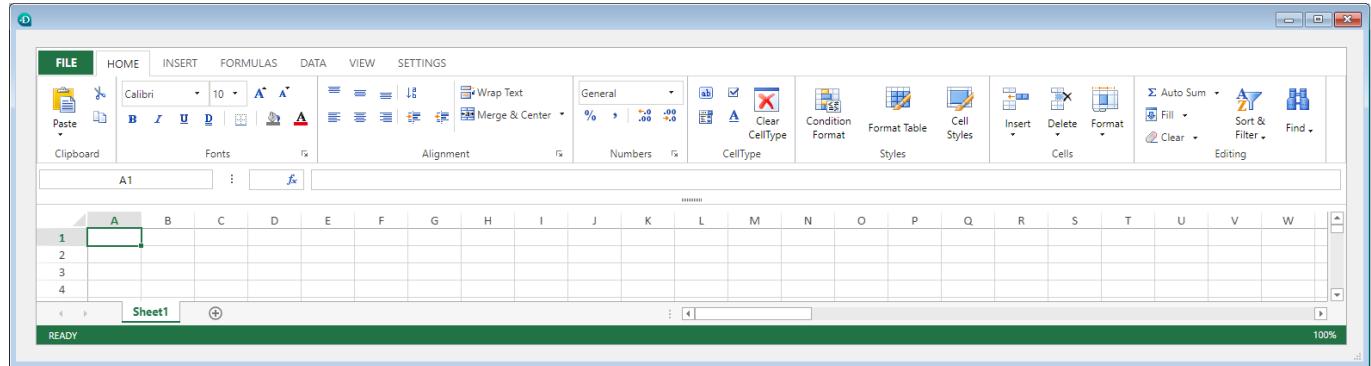
Para una descripción detallada de las funcionalidades de este depurador, consulte la documentación que ofrece el motor de renderizado web.

## Propiedades soportadas

[B](#)[Estilo del borde](#) - [Inferior](#) - [Clase](#) - [Menú contextual](#) - [Altura](#) - [Dim. horizontal](#) - [Izquierda](#) - [Método](#) - [Nombre del objeto](#) - [Progresión](#) - [Derecha](#) - [Arriba](#) - [Tipo](#) - [URL](#) - [Utilizar el motor de renderizado Web integrado](#) - [Variable o expresión](#) - [Dim. vertical](#) - [Visibilidad](#) - [Ancho](#)

# 4D View Pro area

4D View Pro le permite insertar y mostrar un área de hoja de cálculo en sus formularios 4D. Una hoja de cálculo es una aplicación que contiene una cuadrícula de celdas en las que se puede introducir información, ejecutar cálculos o mostrar imágenes.



Una vez que utilice las áreas de 4D View Pro en sus formularios, podrá importar y exportar documentos de hojas de cálculo.

## Utilizar las áreas 4D View Pro

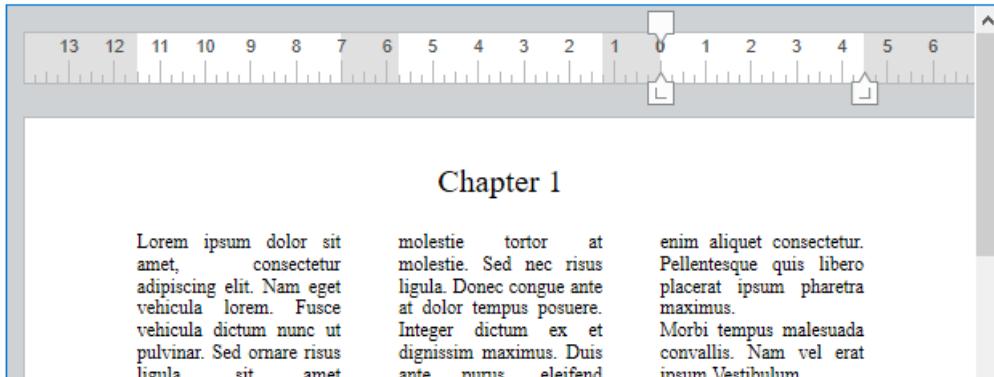
Las áreas 4D View Pro están documentadas en [la sección 4D View Pro](#).

## Propiedades soportadas

Estilo del borde - Abajo - Class - [Altura](properties\_CoordinatesAndSizing. md#height) - Dim. horizontal - Izquierda - Método - [Nombre del objeto](properties\_Object. md#object-name) - Derecha - Mostrar barra de formulario - Tipo - [Interfaz usuario](properties\_Appearance. md#user-interface) - Dim. vertical - Visibilidad - Ancho

# 4D Write Pro area

4D Write Pro ofrece a los usuarios de 4D una herramienta avanzada de procesamiento de textos, totalmente integrada a su aplicación 4D. Con 4D Write Pro, puede escribir correos electrónicos y/o cartas preformateadas que contengan imágenes, una firma escaneada, texto formateado y marcadores de posición para variables dinámicas. También puede crear facturas o informes de forma dinámica, incluyendo texto e imágenes con formato.



## Utilizar las áreas 4D Write Pro

Las áreas 4D Write Pro están documentadas en el manual [4D Write Pro](#).

## Propiedades soportadas

Auto Spellcheck - Border Line Style - Bottom - Class - Context Menu - Draggable - Droppable - Enterable - Focusable - Height - Hide focus rectangle - Horizontal Scroll Bar - Horizontal Sizing - Keyboard Layout - Left - Method - Object Name - Print Variable Frame - Resolution - Right - Selection always visible - Show background - Show footers - Show headers - Show hidden characters - Show horizontal ruler - Show HTML WYSIWYG - Show page frame - Show references - Show vertical ruler - Type - Vertical Sizing - Vertical Scroll Bar - View mode - Visibility - Width - Zoom

# Lista de propiedades JSON

En esta página encontrará una lista completa de todas las propiedades de los objetos ordenadas por su nombre JSON. Haga clic en el nombre de una propiedad para acceder a su descripción detallada.

En el capítulo "Propiedades de los objetos de formulario", las propiedades se ordenan en función de los nombres y temas de la lista de propiedades.

a - b - c - d - e - f - g - h - i - j - k - l - m - n - p - r - s - t - u - v - w - z

Propiedad	Descripción	Valores posibles
a		
<a href="#">action</a>	Acción típica a ejecutar.	El nombre de una acción estándar válida.
<a href="#">allowFontColorPicker</a>	Permite mostrar el selector de fuentes sistema o el selector de colores para editar los atributos de los objetos	true, false (por defecto)
<a href="#">alternateFill</a>	Permite definir un color de fondo diferente para las líneas o columnas impares de un list box.	Todos los valores css; "transparent"; "automatic"; "automaticAlternate"
<a href="#">automaticInsertion</a>	Permite añadir automáticamente un valor a una lista cuando un usuario introduce un valor que no está en la lista de elección asociada al objeto.	true, false
b		
<a href="#">booleanFormat</a>	Indica sólo dos valores posibles.	true, false
<a href="#">borderRadius</a>	El valor del radio para los rectángulos redondos.	mínimo: 0
<a href="#">borderStyle</a>	Permite definir un estilo estándar para el borde del objeto.	"system", "none", "solid", "dotted", "raised", "sunken", "double"
<a href="#">bottom</a>	Posiciona un objeto en la parte inferior (centrado).	mínimo: 0
c		
<a href="#">choiceList</a>	Una lista de opciones asociadas a un objeto	Una lista de selección
<a href="#">class</a>	Una lista de palabras separadas por espacios que se utilizan como selectores de clase en los archivos css.	Una lista de nombres de clases
<a href="#">columnCount</a>	Número de columnas.	mínimo: 1
<a href="#">columns</a>	Una colección de columnas list box	Colección de objetos columna con propiedades de columna definidas
<a href="#">contextMenu</a>	Ofrece al usuario acceso a un menú contextual estándar en el área seleccionada.	"automatic", "none"
<a href="#">continuousExecution</a>	Designa si se ejecuta o no el método de un objeto mientras el usuario	true, false

Propiedad	Descripción	Valores posibles
<code>controlType</code>	Especifica cómo debe representarse el valor en una celda del list box.	"input", "checkbox" (para las columnas booleanas / numéricas), "automatic", "popup" (sólo para columnas booleanas)
<code>currentItemSource</code>	El último elemento seleccionado en un list box.	Expresión del objeto
<code>currentItemPositionSource</code>	La posición del último elemento seleccionado en un list box.	Expresión numérica
<code>customBackgroundPicture</code>	Define la imagen que se dibujará en el fondo de un botón.	Ruta relativa en sintaxis POSIX. Debe utilizarse junto con la opción "Personalizado" de la propiedad "Style".
<code>customBorderX</code>	Define el tamaño (en píxeles) de los márgenes horizontales internos de un objeto. Debe utilizarse con la opción "Personalizado" de la propiedad "Style".	mínimo: 0
<code>customBorderY</code>	Define el tamaño (en píxeles) de los márgenes verticales internos de un objeto. Debe utilizarse con la opción "Personalizado" de la propiedad "Style".	mínimo: 0
<code>customOffset</code>	Define un valor de desplazamiento personalizado en píxeles. Debe utilizarse con la opción "Personalizado" de la propiedad "Style".	mínimo: 0
<code>customProperties</code>	Propiedades avanzadas (si las hay)	Cadena JSON o cadena codificada en base64
d		
<code>dataSource</code> (objects) <code>dataSource</code> (subforms) <code>dataSource</code> (array list box) <code>dataSource</code> (Collection or entity selection list box) <code>dataSource</code> (list box column) <code>dataSource</code> (hierarchical list box)	Indica el origen de los datos.	Una variable 4D, un nombre de campo o una expresión del lenguaje compleja arbitraria.
<code>dataSourceTypeHint</code> (objects) <code>dataSourceTypeHint</code> (list box column, drop-down list)	Indica el tipo de variable.	"integer", "boolean", "number", "picture", "text", date", "time", "arrayText", "arrayDate", "arrayTime", "arrayNumber", "collection", "object", "undefined"
<code>dateFormat</code>	Controls the way times appear when displayed or printed. Sólo debe seleccionarse entre los formatos integrados en 4D.	"systemShort", "systemMedium", "systemLong", "iso8601", "rfc822", "short", "shortCentury", "abbreviated", "long", "blankIfNull" (puede combinarse con otros valores posibles)
<code>defaultButton</code>	Modifica la apariencia de un botón para indicar la opción recomendada al usuario.	true, false
<code>defaultValue</code>	Define un valor o un sello que se introduce por defecto en un objeto	Cadena o "#D", "#H", "#N"

Propiedad	Descripción	Valores posibles
<code>deletableInList</code>	Especifica si el usuario puede eliminar subregistros en un subformulario listado	true, false
<code>detailForm</code> (list box) <code>detailForm</code> (subform)	Asocia un formulario detallado con un subformulario listado.	Nombre (cadena) de la tabla o formulario proyecto, una ruta POSIX (cadena) a un archivo .json que describa el formulario, o un objeto que describa el formulario
<code>display</code>	El objeto se dibuja o no en el formulario.	true, false
<code>doubleClickInEmptyAreaAction</code>	Acción a realizar en caso de doble clic en una línea vacía de un subformulario listado.	"addSubrecord" o "" to do nothing
<code>doubleClickInRowAction</code> (list box) <code>doubleClickInRowAction</code> (subform)	Acción a realizar en caso de doble clic en un registro.	"editSubrecord", "displaySubrecord"
<code>dpi</code>	Resolución de la pantalla para el contenido del área 4D Write Pro.	0=automatic, 72, 96
<code>dragging</code>	Activa la función de arrastrar.	"none", "custom", "automatic" (excluyendo lista, list box)
<code>dropping</code>	Activa la función de soltar.	"none", "custom", "automatic" (excluyendo lista, list box)
e		
<code>editable</code>	Indica si los usuarios pueden introducir valores en el objeto.	true, false
<code>enterableInList</code>	Indica si los usuarios pueden modificar los datos del registro directamente en el subformulario listado.	true, false
<code>entryFilter</code>	Asocia un filtro de entrada con el objeto o las celdas de la columna. Esta propiedad no es accesible si la propiedad Enterable no está activada.	Texto para acotar las entradas
<code>events</code>	Lista de todos los eventos seleccionados para el objeto o el formulario	Colección de nombres de eventos, por ejemplo ["onClick", "onDataChange"...].
<code>excludedList</code>	Permite definir una lista cuyos valores no pueden introducirse en la columna.	Una lista de valores a excluir.
f		
<code>fill</code>	Define el color de fondo de un objeto.	Todo valor CSS, "transparent", "automatic"
<code>focusable</code>	Indica si el objeto puede tener el foco (y por lo tanto puede ser activado por el teclado, por ejemplo)	true, false
<code>fontFamily</code>	Especifica el nombre de la familia de fuentes utilizada en el objeto.	Nombre de la familia de fuentes CSS

<b>fontSize</b> Propiedad	Define el tamaño de la fuente en puntos cuando no se selecciona ningún tema de fuente	mínimo: 0 Valores posibles
<b>fontStyle</b>	Hace que el texto seleccionado se incline ligeramente hacia la derecha.	"normal", "italic"
<b>fontTheme</b>	Establece el estilo automático	"normal", "main", "additional"
<b>fontWeight</b>	Ajusta el texto seleccionado para que aparezca más oscuro y pesado.	"normal", "bold"
<b>footerHeight</b>	Sirve para fijar la altura de la línea	patrón $(\text{\$d+})(\text{pl em})? \text{\$}$ (decimal positivo + px/em )
<b>frameDelay</b>	Permite recorrer el contenido del botón de imagen a la velocidad especificada (en ticks).	mínimo: 0
<b>g</b>		
<b>graduationStep</b>	Medición de la visualización de la escala.	mínimo: 0
<b>h</b>		
<b>header</b>	Define el encabezado de una columna list box	Objeto con propiedades "text", "name", "icon", "dataSource", "fontWeight", "fontStyle", "tooltip"
<b>headerHeight</b>	Sirve para fijar la altura de la línea	patrón $^(\text{\$d+})(\text{px em})? \text{\$}$ (decimal positivo + px/em )
<b>height</b>	Designa el tamaño vertical de un objeto	mínimo: 0
<b>hideExtraBlankRows</b>	Desactiva la visibilidad de las líneas vacías adicionales.	true, false
<b>hideFocusRing</b>	Oculta el rectángulo de selección cuando el objeto tiene el foco.	true, false
<b>hideSystemHighlight</b>	Sirve para especificar la ocultación de los registros resaltados en el list box.	true, false
<b>highlightSet</b>	cadena	Nombre del conjunto.
<b>horizontalLineStroke</b>	Define el color de las líneas horizontales de un list box (gris por defecto).	Todo valor CSS, "transparent", "automatic"
<b>i</b>		
<b>icon</b>	El nombre de la ruta de la imagen utilizada para los botones, casillas de selección, botones de radio y encabezados de list box.	Ruta relativa o filesystem en sintaxis POSIX.
<b>iconFrames</b>	Define el número exacto de estados presentes en la imagen.	mínimo: 1
<b>iconPlacement</b>	Designa la ubicación de un ícono en relación con el objeto formulario.	"ninguno", "izquierda", "derecha"
<b>k</b>		
<b>keyboardDialect</b>	Para asociar una disposición de teclado específica a una entrada.	Una cadena de código de teclado, por ejemplo, "ar-ma"
<b>l</b>		

<code>labels</code>	Propiedad Una lista de valores que se utilizarán como etiquetas de control de pestañas	ej.: "a", "b", "c"… Valores posibles…
<code>labelsPlacement</code> (objects) <code>labelsPlacement</code> (control de pestañas)	Especifica la ubicación del texto mostrado de un objeto.	"none", "top", "bottom", "left", "right"
<code>layoutMode</code>	Modo de visualización del documento 4D Write Pro en el área del formulario.	"page", "draft", "embedded"
<code>left</code>	Posiciona un objeto a la izquierda.	mínimo: 0
<code>list</code> , ver <code>choiceList</code>	Una lista de opciones asociada a una lista jerárquica	Una lista de selección
<code>listboxType</code>	La fuente de datos del list box.	"array", "currentSelection", "namedSelection", "collection"
<code>listForm</code>	Formulario listado a utilizar en el subformulario.	Nombre (cadena) de la tabla o formulario proyecto, una ruta POSIX (cadena) a un archivo .json que describa el formulario, o un objeto que describa el formulario
<code>lockedColumnCount</code>	Número de columnas que deben permanecer permanentemente en la parte izquierda de un list box.	mínimo: 0
<code>loopBackToFirstFrame</code>	Las imágenes se muestran en un bucle continuo.	true, false
m		
<code>max</code>	El valor máximo permitido. En el caso de los steppers numéricos, estas propiedades representan segundos cuando el objeto está asociado a un valor de tipo hora y se ignoran cuando está asociado a un valor de tipo fecha.	mínimo: 0 (para los tipos de datos numéricos)
<code>maxWidth</code>	Designa el mayor tamaño permitido para las columnas list box.	mínimo: 0
<code>metaSource</code>	Un objeto meta que contiene parámetros de estilo y selección.	Una expresión de objeto
<code>method</code>	Un nombre de método proyecto.	El nombre de un método proyecto existente
<code>methodsAccessibility</code>	Qué métodos 4D se pueden llamar desde un área web	"none" (por defecto), "all"
<code>min</code>	El valor mínimo permitido. En el caso de los steppers numéricos, estas propiedades representan segundos cuando el objeto está asociado a un valor de tipo hora y se ignoran cuando está asociado a un valor de tipo fecha.	mínimo: 0 (para los tipos de datos numéricos)
<code>minWidth</code>	Designa el menor tamaño permitido para las columnas list box.	mínimo: 0
<code>movableRows</code>	Autoriza el desplazamiento de líneas durante la ejecución.	true, false

<code>multilinea</code>	Maneja contenidos multilínea. Descripción	"yes", "no", "automatic" Valores posibles
n		
<code>name</code>	El nombre del objeto formulario. (Opcional para el formulario)	Todo nombre que no pertenezca a un objeto ya existente
<code>numberFormat</code>	Controla la forma en que aparecen los campos alfanuméricos y las variables cuando se muestran o imprimen.	Números (incluyendo un punto decimal o un signo menos si es necesario)
p		
<code>imagen</code>	El nombre de la ruta de la imagen para los botones de imagen, los menús emergentes de imagen o las imágenes estáticas	Ruta relativa o del sistema de archivos en sintaxis POSIX, o "var: <variableName>" para una variable tipo imagen.
<code>pictureFormat</code> (input, list box column o footer) <code>pictureFormat</code> (static picture)	Controla la apariencia de las imágenes al mostrarlas o imprimirlas.	"truncatedTopLeft", "scaled", "truncatedCenter", "tiled", "proportionalTopLeft" (excluyendo imágenes estáticas), "proportionalCenter"(excluyendo imágenes estáticas)
<code>placeholder</code>	Desenfoca el texto cuando el valor de la fuente de datos está vacío.	Texto que debe estar en gris.
<code>pluginAreaKind</code>	Describe el tipo de plug-in.	El tipo de plug-in.
<code>popupPlacement</code>	Permite mostrar un símbolo que aparece como un triángulo en el botón, que indica que hay un menú emergente adjunto.	"None", "Linked", "Separated"
<code>printFrame</code>	Modo de impresión para objetos cuyo tamaño puede variar de un registro a otro en función de su contenido	"fixed", "variable", (subformulario únicamente) "fixedMultiple"
<code>progressSource</code>	Un valor entre 0 y 100, que representa el porcentaje de finalización de la carga de la página en el área web. Actualizado automáticamente por 4D, no puede ser modificado manualmente.	mínimo: 0
r		
<code>radioGroup</code>	Permite utilizar los botones de radio en conjuntos coordinados: sólo se puede seleccionar un botón a la vez en el conjunto.	Nombre del grupo radio
<code>requiredList</code>	Permite definir una lista en la que sólo se pueden insertar determinados valores.	Una lista de valores obligatorios.
<code>redimensionable</code>	Designa si el tamaño de un objeto puede ser modificado por el usuario.	"true", "false"
<code>resizingMode</code>	Specifies if a list box column should be automatically resized	"rightToLeft", "legacy"
<code>right</code>	Posiciona un objeto a la derecha.	mínimo: 0
<code>rowControlSource</code>	Un array 4D que define las líneas del list box.	Array

Propiedad	Descripción	Valores posibles
<code>rowFillSource</code> (array list box) <code>rowFillSource</code> (selection o collection list box)	El nombre de un array o expresión para aplicar un color de fondo personalizado a cada línea de un list box.	El nombre de un array o expresión.
<code>rowHeight</code>	Define la altura de las líneas del list box.	Valor CSS la unidad "em" o "px" (por defecto)
<code>rowHeightAuto</code>	booleano	"true", "false"
<code>rowHeightAutoMax</code>	Designa la mayor altura permitida para las líneas del list box.	Valor CSS la unidad "em" o "px" (por defecto). mínimo: 0
<code>rowHeightAutoMin</code>	Designa la menor altura permitida para las líneas del list box.	Valor CSS la unidad "em" o "px" (por defecto). mínimo: 0
<code>rowHeightSource</code>	Un array que define diferentes alturas para las líneas de un list box.	Nombre de una variable array 4D.
<code>rowStrokeSource</code> (array list box) <code>rowStrokeSource</code> (selection o collection/entity selection list box)	Un array o expresión para gestionar los colores de las líneas.	Nombre del array o expresión.
<code>rowStyleSource</code> (array list box) <code>rowStyleSource</code> (selection o collection/entity selection list box)	Un array o expresión para gestionar los estilos de las líneas.	Nombre del array o expresión.
s		
<code>saveAs</code> (list box column) <code>saveAs</code> (drop-down list)	El tipo de contenido a guardar en el campo o variable asociado al objeto formulario	"value", "reference"
<code>scrollbarHorizontal</code>	Una herramienta que permite al usuario desplazar el área de visualización hacia la izquierda o la derecha.	"visible", "hidden", "automatic"
<code>scrollbarVertical</code>	Una herramienta que permite al usuario mover el área de visualización hacia arriba o hacia abajo.	"visible", "hidden", "automatic"
<code>selectedItemsSource</code>	Colección de los elementos seleccionados en un list box.	Expresión de la colección
<code>selectionMode</code> (hierarchical list) <code>selectionMode</code> (list box) <code>selectionMode</code> (subform)	Permite la selección de múltiples registros/líneas.	"multiple", "single", "none"
<code>shortcutAccel</code>	Especifica el sistema a utilizar, Windows o Mac.	true, false
<code>shortcutAlt</code>	Designa la tecla Alt	true, false
<code>shortcutCommand</code>	Designa la tecla Comando (macOS)	true, false
<code>shortcutControl</code>	Designa la tecla Control (Windows)	true, false
<code>shortcutKey</code>	La letra o el nombre de una tecla de significado especial.	"[F1]" -> "[F15]", "[Return]", "[Enter]", "[Backspace]", "[Tab]", "[Esc]", "[Del]", "[Home]", "[End]", "[Help]", "[Page up]", "[Page down]", "[Left arrow]", "[Right arrow]"

Propiedad	Descripción	<code>up</code> , <code>page down</code> , <code>left arrow</code> , <code>right arrow</code> [arriba], [abajo], [izquierda], [derecha]
<code>shortcutShift</code>	Designa la tecla Mayús	true, false
<code>showFooters</code>	Muestra u oculta los pies de página de las columnas.	true, false
<code>showGraduations</code>	Muestra/Oculta las graduaciones junto a las etiquetas.	true, false
<code>showHeaders</code>	Muestra u oculta los encabezados de las columnas.	true, false
<code>showHiddenChars</code>	Muestra/oculta los caracteres invisibles.	true, false
<code>showHorizontalRuler</code>	Muestra/oculta la regla horizontal cuando la vista del documento está en modo vista Página	true, false
<code>showHTMLWysiwyg</code>	Activa/desactiva la vista HTML WYSIWYG	true, false
<code>showPageFrames</code>	Muestra/oculta el marco de página cuando la vista del documento está en modo vista Página	true, false
<code>showReferences</code>	Muestra todas las expresiones 4D insertadas en el documento de 4D Write Pro como <i>referencias</i>	true, false
<code>showSelection</code>	Mantiene la selección visible dentro del objeto después de haber perdido el foco	true, false
<code>showVerticalRuler</code>	Muestra/oculta la regla vertical cuando la vista del documento está en modo vista Página	true, false
<code>singleClickEdit</code>	Permite el paso directo al modo de edición.	true, false
<code>sizingX</code>	Especifica si el tamaño horizontal de un objeto debe ser movido o redimensionado cuando un usuario cambia el tamaño del formulario.	"grow", "move", "fixed"
<code>sizingY</code>	Especifica si el tamaño vertical de un objeto debe ser movido o redimensionado cuando un usuario cambia el tamaño del formulario.	"grow", "move", "fixed"
<code>sortable</code>	Permite ordenar los datos de las columnas haciendo clic en el encabezado.	true, false
<code>spellcheck</code>	Activa la corrección ortográfica para el objeto	true, false
<code>splitterMode</code>	Cuando un objeto splitter tiene esta propiedad, los otros objetos a su derecha (splitter vertical) o debajo de él (separador horizontal) son empujados al mismo tiempo que el splitter, sin parar.	"grow", "move", "fixed"
<code>startPoint</code>	Punto de partida para dibujar un objeto de línea (sólo disponible en	"bottomLeft", topLeft"

Propiedad	Descripción (en formato JSON).	Valores posibles
<code>staticColumnCount</code>	Número de columnas que no se pueden mover durante la ejecución.	mínimo: 0
<code>step</code>	Intervalo mínimo aceptado entre los valores durante el uso. Para los steppers numéricos, esta propiedad representa los segundos cuando el objeto está asociado a un valor de tipo hora y los días cuando está asociado a un valor de tipo fecha.	mínimo: 1
<code>storeDefaultStyle</code>	Almacenar las etiquetas de estilo con el texto, incluso si no se ha realizado ninguna modificación	true, false
<code>stroke</code> (text) <code>stroke</code> (lines) <code>stroke</code> (list box)	Especifica el color de la fuente o línea utilizada en el objeto.	Todo valor CSS, "transparent", "automatic"
<code>strokeDashArray</code>	Describe el tipo de línea punteada como una secuencia de puntos blancos y negros	Arrays numéricos o cadenas
<code>strokeWidth</code>	Designa el grosor de una línea.	Un entero o 0 para el ancho más pequeño en un formulario impreso
<code>style</code>	Permite definir el aspecto general del botón. Para más información, consulte Estilo de los botones.	"regular", "flat", "toolbar", "bevel", "roundedBevel", "gradientBevel", "texturedBevel", "office", "help", "circular", "disclosure", "roundedDisclosure", "custom"
<code>styledText</code>	Permite utilizar los estilos específicos en el área seleccionada.	true, false
<code>switchBackWhenReleased</code>	Muestra la primera imagen todo el tiempo, excepto cuando el usuario hace clic en el botón. Muestra la segunda imagen hasta que se suelta el botón del ratón.	true, false
<code>switchContinuously</code>	Permite al usuario mantener pulsado el botón del ratón para mostrar las imágenes de forma continua (es decir, como una animación).	true, false
<code>switchWhenRollover</code>	Modifica el contenido del botón de la imagen cuando el cursor del ratón pasa por encima. La imagen inicial se muestra cuando el cursor sale del área del botón.	true, false
<code>t</code>		
<code>tabla</code>	Tabla a la que pertenece el subformulario Lista (si lo hay).	Nombre de tabla 4D, o ""
<code>texto</code>	El título del objeto formulario	Todo texto
<code>textAlign</code>	Ubicación horizontal del texto dentro del área que lo contiene.	"automatic", "right", "center", "justify", "left"
<code>textAngle</code>	Modifica la orientación (rotación) del área de texto.	0, 90, 180, 270

<code>textDecoration</code>	Hace que el texto seleccionado tenga una línea por debajo.	"normal", "underline" Valores posibles
<code>textFormat</code>	Controla la forma en que aparecen los campos alfanuméricos y las variables cuando se muestran o imprimen.	"#### #####", "(###) #### ####", "### #### #####", "### ## ######", "00000", formatos personalizados
<code>textPlacement</code>	Ubicación relativa del título del botón en relación con el ícono asociado.	"left", "top", "right", "bottom", "center"
<code>threeState</code>	Permite que un objeto de casilla de selección acepte un tercer estado.	true, false
<code>timeFormat</code>	Controla la forma en que aparecen las fechas cuando se muestran o imprimen. Sólo debe seleccionarse entre los formatos integrados en 4D.	"systemShort", "systemMedium", "systemLong", "iso8601", "hh_mm_ss", "hh_mm", "hh_mm_am", "mm_ss", "HH_MM_SS", "HH_MM", "MM_SS", "blankIfNull" (puede combinarse con otros valores posibles)
<code>truncateMode</code>	Controla la visualización de los valores cuando las columnas del list box son demasiado estrechas para mostrar todo su contenido.	"withEllipsis", "none"
<code>type</code>	Obligatorio. Designa el tipo de datos del objeto formulario.	"text", "rectangle", "groupBox", "tab", "line", "button", "checkbox", "radio", "dropdown", "combo", "webArea", "write", "subform", "plugin", "splitter", "buttonGrid", "progress", "ruler", "spinner", "stepper", "list", "pictureButton", "picturePopup", "listbox", "input", "view"
<code>tooltip</code>	Ofrece a los usuarios información adicional sobre un campo.	Información adicional para ayudar al usuario
<code>top</code>	Posiciona un objeto en la parte superior (centrado).	mínimo: 0
u		
<code>urlSource</code>	Designa la URL cargada o que está siendo cargada por el área web asociada.	Una URL.
<code>useLastFrameAsDisabled</code>	Permite definir la última miniatura como la que se mostrará cuando el botón esté desactivado.	true, false
<code>userInterface</code>	Interfaz del área 4D View Pro.	"none" (por defecto), "ribbon", "toolbar"
v		
<code>values</code>	Lista de valores por defecto para una columna de listbox array	ej.: "A","B","42"...
<code>variableCalculation</code>	Permite realizar cálculos matemáticos.	"none", "minimum", "maximum", "sum", "count", "average", "standardDeviation", "variance", "sumSquare"
<code>verticalAlign</code>	Ubicación vertical del texto dentro del área que lo contiene.	"automatic", "top", "middle", "bottom"
<code>verticalLineStroke</code>	Define el color de las líneas verticales de un list box (gris por	Todo valor CSS, "transparent", "automatic"

Propiedad	Descripción	Valores posibles
<code>visibility</code>	Permite ocultar el objeto en el entorno de la aplicación.	"visible", "hidden", "selectedRows", "unselectedRows"
w		
<code>webEngine</code>	Permite elegir entre dos motores de renderizado para el área web, dependiendo de las particularidades de la aplicación.	"embedded", "system"
<code>ancho</code>	Designa el tamaño horizontal de un objeto	mínimo: 0
<code>withFormulaBar</code>	Gestiona la visualización de una barra de fórmulas con la interfaz de la barra de herramientas en el área 4D View Pro.	true, false
<code>wordwrap</code>	Gestiona la visualización del contenido cuando supera el ancho del objeto.	"automatic" (excluyendo list box), "normal", "none"
z		
<code>zoom</code>	Porcentaje de zoom para mostrar el área 4D Write Pro	numérico (mínimo=0)

# Acción

---

## Arrastrable

Controle si el usuario puede arrastrar el objeto y cómo. Por defecto, no se permite ninguna operación de arrastre.

Hay dos modos de arrastrar disponibles:

- Personalizado: en este modo, toda operación de arrastrar realizada en el objeto dispara el evento formulario `On Begin Drag` en el contexto del objeto. A continuación, se gestiona la acción de soltar mediante un método. En modo personalizado, básicamente toda la operación de arrastrar y soltar es manejada por el programador. Este modo le permite implementar cualquier interfaz basada en la función de arrastrar y soltar, incluidas las interfaces que no necesariamente transportan datos, sino que pueden realizar cualquier acción como abrir archivos o activar un cálculo. Este modo se basa en una combinación de propiedades, eventos y comandos específicos del tema [Portapapeles](#).
- Automático: en este modo, 4D copia el texto o las imágenes directamente desde el objeto formulario. Puede utilizarse en la misma área 4D, entre dos áreas 4D o entre 4D y otra aplicación. Por ejemplo, el arrastre (y soltado) automático permite copiar un valor entre dos campos sin necesidad programación:



En este modo, el evento de formulario `On Begin Drag`

NO se genera. Si quiere "forzar" el uso del arrastre personalizado mientras está activado el arrastre automático, mantenga presionada la tecla Alt (Windows) o Opción (macOS) durante la acción. Esta opción no está disponible para las imágenes.

Para más información, consulte [Arrastrar y soltar](#) en el manual *Lenguaje 4D*.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
dragging	texto	"none" (por defecto), "custom", "automatic" (excluyendo list box)

## Objetos soportados

[Áreas 4D Write Pro](#) - [Entrada](#) - [Lista jerárquica](#) - [List Box](#) - [Área de Plug-in](#)

Ver también

[Soltable](#)

---

## Soltable

Controla si el objeto puede ser el destino de una operación de arrastrar y soltar y cómo hacerlo.

Hay dos modos de soltar disponibles:

- Personalizado: en este modo, cualquier operación de soltar realizada en el objeto activa los eventos formulario `On Drag Over` y `On Drop` en el contexto del objeto. A continuación, se gestiona la acción de soltar mediante un método. En modo personalizado, básicamente toda la operación de arrastrar y soltar es manejada por el programador. Este modo le permite implementar cualquier interfaz basada en la función de arrastrar y soltar, incluidas las interfaces que no necesariamente transportan datos, sino que pueden realizar cualquier acción como abrir archivos o activar

un cálculo. Este modo se basa en una combinación de propiedades, eventos y comandos específicos del tema **Portapapeles**.

- Automático: en este modo, 4D gestiona automáticamente, si es posible, la inserción de los datos arrastrados de tipo texto o imagen que se sueltan sobre el objeto (los datos se pegan en el objeto). Los eventos `On Drag Over` y `On Drop` NO se generan. Por otra parte, se generan los eventos `On After Edit` (durante el soltar) y `On Data Change` (cuando el objeto pierde el foco).

Para más información, consulte [Arrastrar y soltar](#) en el manual *Lenguaje 4D*.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
dropping	texto	"none" (por defecto), "custom", "automatic" (excluyendo list box)

#### Objetos soportados

[Áreas 4D Write Pro](#) - [Botón](#) - [Entrada](#) - [Lista jerárquica](#) - [List Box](#) - [Área de Plug-in](#)

#### Ver también

[Arrastrable](#)

## Ejecutar método objeto

Cuando esta opción está activada, el método objeto se ejecuta con el evento `On Data Change` en el mismo momento en que el usuario cambia el valor del indicador. Cuando la opción está desactivada, el método se ejecuta tras la modificación.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
continuousExecution	booleano	true, false

#### Objetos soportados

[Indicador de progreso](#) - [Regla](#) - [Stepper](#)

## Método

Referencia de un método adjunto al objeto. Los métodos de objeto generalmente "gestionan" el objeto mientras el formulario se muestra o se imprime. No llame a un método objeto, 4D lo llama automáticamente cuando un evento implica el objeto al que el método objeto está asociado.

Se soportan varios tipos de referencias de métodos:

- una ruta de archivo de método objeto estándar, es decir, que utilice el siguiente patrón:  
`ObjectMethods/ObjectName.4dm`  
... donde `ObjectName` es el [nombre del objeto](#). Este tipo de referencia indica que el archivo del método se encuentra en la ubicación por defecto ("sources/forms/formName/ObjectMethods/"). En este caso, 4D maneja automáticamente el método objeto cuando se ejecutan operaciones en el objeto formulario (renombrar, duplicar, copiar/pegar...)
- un nombre de método proyecto: nombre de un método proyecto existente sin extensión de archivo, es decir:  
`myMethod` En este caso, 4D no soporta automáticamente las operaciones objeto.
- una ruta de acceso al archivo del método personalizado que incluya la extensión .4dm, por ejemplo:  
`../../CustomMethods/myMethod.4dm` También puede utilizar un sistema de archivos:

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
method	texto	Ruta de archivo estándar o personalizada del método objeto o nombre del método proyecto

## Objetos soportados

[Área 4D View Pro](#) - [Área 4D Write Pro](#) - [Botón](#) - [Rejilla de botones](#) - [Casilla de selección](#) - [Combo Box](#) - [Lista desplegable](#) - [Formularios](#) - [Lista jerárquica](#) - [Entrada](#) - [List Box](#) - [Columna List Box](#) - [Botón imagen](#) - [Menú emergente de imagen](#) - [Área de plugins](#) - [Indicadores de progreso](#) - [Botón radio](#) - [Regla](#) - [Spinner](#) - [Splitter](#) - [Stepper](#) - [Subformulario](#) - [Control de pestañas](#) - [Área web](#)

---

## Líneas desplazables

### List box de tipo array

Autoriza el desplazamiento de líneas durante la ejecución. Esta opción está seleccionada por defecto. No está disponible para los [list box de tipo selección](#) ni para los [list box en modo jerárquico](#).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
movableRows	booleano	true, false

## Objetos soportados

### [List Box](#)

---

## Multi-seleccionable

Permite la selección de múltiples registros/opciones en una [lista jerárquica](#).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
selectionMode	texto	"multiple", "single", "none"

## Objetos soportados

### [Lista jerárquica](#)

---

## Ordenable

Permite ordenar los datos de las columnas haciendo clic en un encabezado [listbox](#). Esta opción está seleccionada por defecto. Los arrays de tipo imagen (columnas) no pueden ordenarse utilizando esta función.

En los list box basados en una selección de registros, sólo está disponible la función de ordenación estándar:

- Cuando la fuente de datos es *Selección actual*,

- Con columnas asociadas a campos (de tipo Alfa, Número, Fecha, Hora o Booleano).

En otros casos (list box basados en selecciones temporales, columnas asociadas a expresiones), la función de ordenación estándar no está disponible. Una ordenación estándar del list box cambia el orden de la selección actual en la base de datos. Sin embargo, los registros resaltados y el registro actual no se modifican. Una ordenación estándar sincroniza todas las columnas del list box, incluidas las columnas calculadas.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
sortable	booleano	true, false

#### Objetos soportados

##### List Box

---

## Acción estándar

Actividades típicas que deben realizar por los objetos activos (*por ejemplo*, permitir que el usuario acepte, cancele o elimine registros, se desplace entre registros o de una página a otra en un formulario de varias páginas, etc.) han sido predefinidas por 4D como acciones estándar. Se describen con detalle en la sección [Acciones estándar](#) de la *manual de Diseño*.

Puede asignar al mismo tiempo una acción estándar y un método proyecto de un objeto. En este caso, la acción estándar suele ejecutarse después del método y 4D utiliza esta acción para activar/desactivar el objeto según el contexto actual. Cuando se desactiva un objeto, no se puede ejecutar el método proyecto asociado.

También puede definir esta propiedad utilizando el comando `OBJECT SET ACTION`.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
action	cadena	El nombre de una <a href="#">acción estándar válida</a> .

#### Objetos soportados

[Botón](#) - [Rejilla de botones](#) - [Casilla de selección](#) - [Lista desplegable](#) - [List Box](#) - [Botón imagen](#) - [Pop-up Menu imagen](#) - [Pestaña](#)

# Animación

---

## Vuelve a la primera secuencia

Las imágenes se muestran en un bucle continuo. Cuando el usuario llega a la última imagen y vuelve a hacer clic, aparece la primera imagen, y así sucesivamente.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
loopBackToFirstFrame	booleano	true, false

Objetos soportados

[Botón Imagen](#)

---

## Retorna cuando se libera

Muestra la primera imagen todo el tiempo, excepto cuando el usuario hace clic en el botón. Muestra la segunda imagen hasta que se suelta el botón del ratón. Este modo le permite crear un botón de acción con una imagen diferente para cada estado (inactivo y pulsado). Puede utilizar este modo para crear un efecto 3D o mostrar cualquier imagen que represente la acción del botón.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
switchBackWhenReleased	booleano	true, false

Objetos soportados

[Botón Imagen](#)

---

## Desplazamiento continuo en clics

Permite al usuario mantener pulsado el botón del ratón para mostrar las imágenes de forma continua (es decir, como una animación). Cuando el usuario llega a la última imagen, el objeto no vuelve a la primera imagen.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
switchContinuously	booleano	true, false

Objetos soportados

[Botón Imagen](#)

---

## Cambiar cada x ticks

Permite recorrer el contenido del botón de imagen a la velocidad especificada (en ticks). En este modo, se ignoran todas las demás opciones.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
frameDelay	integer	mínimo: 0

Objetos soportados

[Botón Imagen](#)

---

## Cambiar cuando se pasa por encima el cursor

Modifica el contenido del botón de la imagen cuando el cursor del ratón pasa por encima. La imagen inicial se muestra cuando el cursor sale del área del botón.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
switchWhenRollover	booleano	true, false

Objetos soportados

[Botón Imagen](#)

---

## Utilizar el Último cuadro como Desactivado

Permite definir la última miniatura como la que se mostrará cuando el botón esté desactivado. La imagen en miniatura utilizada cuando el botón está desactivado es procesada por separado por 4D: cuando se combina esta opción con "Cambiar continuamente" y "Retroceder en bucle a la primera imagen", la última imagen se excluye de la secuencia asociada al botón y sólo aparece cuando está desactivado.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
useLastFrameAsDisabled	booleano	true, false

Objetos soportados

[Botón Imagen](#)

# Apariencia

---

## Botón por defecto

La propiedad del botón por defecto designa el botón que obtiene el foco inicial en ejecución cuando ningún botón del formulario tiene la propiedad [Focusable](#).

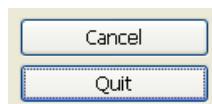
Sólo puede haber un botón por defecto por página de formulario.

Además, en macOS, la propiedad del botón por defecto modifica la apariencia del botón para indicar una "opción recomendada" al usuario. El botón por defecto puede ser diferente del botón enfocado. Los botones por defecto tienen un aspecto azul específico en macOS:



El botón debe tener una altura estándar para obtener la apariencia de botón por defecto.

En Windows, no se soporta el concepto de "elección recomendada": sólo el botón enfocado tiene una apariencia diferente en tiempo de ejecución. Sin embargo, en el editor de formularios de 4D, el botón por defecto se representa con un contorno azul:



### Gramática JSON

Nombre	Tipos de datos	Valores posibles
defaultButton	boolean	true, false

### Objetos soportados

[Botón regular - Botón plano](#)

---

## Ocultar rectángulo de enfoque

Durante la ejecución, un campo o toda área introducible es delimitada por un rectángulo de selección cuando tiene el foco (a través de la tecla Tab o un simple clic). Puede ocultar este rectángulo activando esta propiedad. Ocultar el rectángulo de enfoque puede ser útil en el caso de interfaces específicas.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
hideFocusRing	booleano	true, false

### Objetos soportados

[Áreas 4D Write Pro - Lista jerárquica - Área de entrada - List Box - Sub formulario](#)

---

## Ocultar resaltado selección

### List boxes de tipo selección

Esta propiedad se utiliza para desactivar el resaltado de la selección en los list box.

Cuando esta opción está activada, el resaltado de la selección ya no es visible para las selecciones realizadas en los list box. Las selecciones en sí siguen siendo válidas y funcionan exactamente igual que antes; sin embargo, ya no se representan gráficamente en la pantalla, y tendrá que [definir su apariencia por programación](#).

Por defecto, esta opción no está activa.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
hideSystemHighlight	booleano	true, false

### Objetos soportados

#### [List Box](#)

## Barra de desplazamiento horizontal

Una herramienta de interfaz que permite al usuario desplazar el área de visualización hacia la izquierda o la derecha.

Valores disponibles:

Lista de propiedades	Valor JSON	Descripción
Sí	"visible"	La barra de desplazamiento está siempre visible, incluso cuando no es necesaria (es decir, cuando el tamaño del contenido del objeto es menor que el del marco).
No	"hidden"	La barra de desplazamiento nunca es visible
Automático	"automatic"	La barra de desplazamiento aparece automáticamente cuando es necesario y el usuario puede introducir un texto mayor que el ancho del objeto

Los objetos imagen pueden tener las barras de desplazamiento cuando el formato de visualización de la imagen está definido como "Truncado (no centrado)."

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
scrollbarHorizontal	texto	"visible", "hidden", "automatic"

### Objetos soportados

#### [Lista jerárquica](#) - [Sub formulario](#) - [List Box](#) - [Área de entrada](#) - [Área 4D Write Pro](#)

### Ver también

#### [Barra de desplazamiento vertical](#)

## Resolution

Define la resolución de la pantalla para el contenido del área 4D Write Pro. Por defecto, se define a 72 ppp (macOS),

que es la resolución estándar para los formularios 4D en todas las plataformas. Si se define esta propiedad en 96 ppp, se establecerá un renderizado Windows/Web tanto en plataformas macOS como Windows. Si se define esta propiedad como automática significa que la representación del documento diferirá entre las plataformas macOS y Windows.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
dpi	number	0=automatic, 72, 96

Objetos soportados

Área 4D Write Pro

---

## Mostrar el fondo

Muestra/oculta tanto las imágenes de fondo como el color de fondo.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
showBackground	boolean	true (por defecto), false

Objetos soportados

Área 4D Write Pro

---

## Mostrar los pies de página

Muestra/oculta los pies de página cuando el [modo visualización de la página](#) está definido como "Página".

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
showFooters	boolean	true (por defecto), false

Objetos soportados

Área 4D Write Pro

---

## Mostrar la barra de fórmula

Cuando está activada, la barra de fórmulas es visible debajo de la interfaz de la barra de herramientas en el área 4D View Pro. Si no se selecciona, la barra de fórmulas se oculta.

Esta propiedad sólo está disponible para la interfaz de la [Barra de herramientas](#).

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
withFormulaBar	booléen	true (por defecto), false

Objetos soportados

[Área 4D View Pro](#)

---

## Mostrar los encabezados

Muestra/oculta los encabezados cuando el [modo visualización de la página](#) está definido como "Página".

Gramática JSON

Nombre	Tipos de datos	Valores posibles
showHeaders	boolean	true (por defecto), false

Objetos soportados

[Área 4D Write Pro](#)

---

## Mostrar los caracteres ocultos

Muestra/oculta los caracteres invisibles

Gramática JSON

Nombre	Tipos de datos	Valores posibles
showHiddenChars	boolean	true (por defecto), false

Objetos soportados

[Área 4D Write Pro](#)

---

## Mostrar la regla horizontal

Muestra/oculta la regla horizontal cuando la vista del documento está en modo [Página](#).

Gramática JSON

Nombre	Tipos de datos	Valores posibles
showHorizontalRuler	boolean	true (por defecto), false

Objetos soportados

[Área 4D Write Pro](#)

---

## Mostrar HTML WYSIWIG

Activa/desactiva la vista HTML WYSIWYG, en la que se eliminan los atributos avanzados de 4D Write Pro que no son compatibles con todos los navegadores.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
showHTMLWysiwyg	boolean	true, false (por defecto)

Objetos soportados

Área 4D Write Pro

---

## Mostrar el marco de la página

Muestra/oculta el marco de la página cuando [modo visualización de página](#) está definido como "Página".

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
showPageFrames	boolean	true, false

Objetos soportados

Área 4D Write Pro

---

## Mostrar las referencias

Muestra todas las expresiones 4D insertadas en el documento de 4D Write Pro como *referencias*. Cuando esta opción está desactivada, las expresiones 4D se muestran como *valores*. Por defecto, cuando se inserta un campo o expresión 4D, 4D Write Pro calcula y muestra su valor actual. Seleccione esta propiedad si desea saber qué campo o expresión se muestra. Las referencias de campo o de expresión aparecen entonces en su documento, con un fondo gris.

Por ejemplo, ha insertado la fecha actual junto con un formato, la fecha se muestra:

July 11, 2016

Con la propiedad Mostrar referencias activada, se muestra la referencia:

String(Current date;Internal date long)

Las expresiones 4D se pueden insertar con el comando [ST INSERT EXPRESSION](#).

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
showReferences	boolean	true, false (por defecto)

Objetos soportados

Área 4D Write Pro

---

## Mostrar regla vertical

Muestra/oculta la regla vertical cuando la vista del documento está en modo [Página](#).

Gramática JSON

Nombre	Tipos de datos	Valores posibles
showVerticalRuler	boolean	true (por defecto), false

Objetos soportados

[Área 4D Write Pro](#)

---

## Pestañas

Puede definir la dirección de las pestañas en sus formularios. Esta propiedad está disponible en todas las plataformas, pero sólo puede mostrarse en macOS. Puede elegir colocar los controles de las pestañas en la parte superior (estándar) o en la parte inferior.

Cuando los controles de pestañas con una dirección personalizada se muestran en Windows, vuelven automáticamente a la dirección estándar (superior).

Gramática JSON

Nombre	Tipos de datos	Valores posibles
labelsPlacement	boolean	"top", "bottom"

Objetos soportados

[Pestañas](#)

---

## Interfaz de usuario

Puede añadir una interfaz a las áreas 4D View Pro para permitir a los usuarios finales realizar modificaciones básicas y manipulaciones de datos. 4D View Pro ofrece dos interfaces opcionales a elegir, Cinta y Barra de herramientas.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
userInterface	text	"none" (por defecto), "ribbon", "toolbar"

Objetos soportados

[Área 4D View Pro](#)

Ver también

[guía de referencia 4D View Pro](#)

---

## Barra de desplazamiento vertical

Una herramienta de interfaz que permite al usuario mover el área de visualización hacia arriba y hacia abajo.

Valores disponibles:

Lista de propiedades	Valor JSON	Descripción
Sí	"visible"	La barra de desplazamiento está siempre visible, incluso cuando no es necesaria (es decir, cuando el tamaño del contenido del objeto es menor que el del marco).
No	"hidden"	La barra de desplazamiento nunca es visible
Automático	"automatic"	La barra de desplazamiento aparece automáticamente cuando es necesario (es decir, cuando el tamaño del contenido del objeto es mayor que el del marco)

Los objetos imagen pueden tener las barras de desplazamiento cuando el formato de visualización de la imagen está definido como "Truncado (no centrado)."

Si un objeto de entrada de texto no tiene una barra de desplazamiento, el usuario puede desplazarse por la información utilizando las teclas de flecha.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
scrollbarVertical	texto	"visible", "hidden", "automatic"

## Objetos soportados

[Lista jerárquica](#) - [Sub formulario](#) - [List Box](#) - [Área de entrada](#) - [Área 4D Write Pro](#)

Ver también

[Barra de desplazamiento horizontal](#)

## Modo de visualización

Establece el modo de visualización del documento de 4D Write Pro en el área del formulario. Hay tres valores disponibles:

- Página: el modo de vista más completo, que incluye contornos de página, orientación, márgenes, saltos de página, encabezados y pies de página, etc.
- Borrador: modo borrador con propiedades básicas del documento
- Embedded: modo de vista adecuado para zonas integradas; no muestra márgenes, pies de página, encabezados, marcos de página, etc. Este modo también se puede utilizar para producir una salida de vista similar a la de la web (si también selecciona la [resolución de 96 dpi](#) y las propiedades [Mostrar HTML WYSIWYG](#)).

La propiedad Modo vista sólo se utiliza para la renderización en pantalla. En cuanto a la configuración de la impresión, se utilizan automáticamente reglas de renderización específicas.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
layoutMode	text	"page", "draft", "embedded"

## Objetos soportados

## Zoom

Define el porcentaje de zoom para mostrar el contenido del área 4D Write Pro.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
zoom	número	minimum = 0

Objetos soportados

Área 4D Write Pro

# Fondo y borde

---

## Color de fondo alternado

Permite definir un color de fondo diferente para las líneas o columnas impares de un list box. Por defecto, *Automático* está seleccionado: la columna utiliza el color de fondo alternativo definido en el nivel del list box.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
alternateFill	cadena	todos los valores css; "transparent"; "automatic"; "automaticAlternate"

Objetos soportados

[List Box](#) - [Columna List Box](#)

---

## Color de fondo / Color de relleno

Define el color de fondo de un objeto.

En el caso de un list box, por defecto se selecciona *Automático*: la columna utiliza el color de fondo definido al nivel del list box.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
fill	cadena	un valor css; "transparent"; "automatic"

Objetos soportados

[Lista jerárquica](#) - [List Box](#) - [Columna List Box](#) - [Pie de List Box](#) - [Óvalo](#) - [Rectángulo](#) - [Área de texto](#)

Ver también

[Transparente](#)

---

## Expresión color de fondo

`List box de tipo colección y de tipo selección de entidades`

Una expresión o una variable (no se pueden utilizar variables array) para aplicar un color de fondo personalizado a cada línea del list box. La expresión o la variable se evaluará para cada línea mostrada y debe devolver un valor de color RGB. Para más información, consulte la descripción del comando `OBJECT SET RGB COLORS` en el manual *Lenguaje de 4D*.

También puede definir esta propiedad utilizando el comando `LISTBOX SET PROPERTY` con la constante `lk background color expression`.

Con los list box de tipo colección o selección de entidades, esta propiedad también puede definirse utilizando una [Meta Info Expression](#).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
rowFillSource	cadena	Una expresión que devuelve un valor de color RGB

Objetos soportados

[List Box](#) - [Columna List Box](#)

---

## Estilo del borde

Permite definir un estilo estándar para el borde del objeto.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
borderStyle	texto	"system", "none", "solid", "dotted", "raised", "sunken", "double"

Objetos soportados

[Área 4D View Pro](#) - [Área 4D Write Pro](#) - [Botones](#) - [Rejilla de botones](#) - [Casilla de selección](#) - [Área de entrada](#) - [List Box](#) - [Botón imagen](#) - [Menú emergente de imagen](#) - [Área de plugin](#) - [Indicador de progreso](#) - [Regla](#) - [Spinner](#) - [Stepper](#) - [Subformulario](#) - [Área de texto](#) - [Área Web](#)

---

## Tipo de línea punteada

Describe el tipo de línea punteada como una secuencia de puntos blancos y negros.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
strokeDashArray	arrays numéricos o cadenas	Ej. "6 1" o [6,1] para una secuencia de 6 puntos negros y 1 punto blanco

Objetos soportados

[Rectángulo](#) - [Óvalo](#) - [Línea](#)

---

## Ocultar líneas vacías finales

Controla la visualización de las líneas vacías adicionales añadidas en la parte inferior de un objeto list box. Por defecto, 4D añade esas líneas adicionales para llenar el área vacía:

Days	Values
Monday	5
Tuesday	6
Wednesday	41
Thursday	66
Friday	12
Saturday	2
Sunday	88

Extra blank rows

Puede eliminar estas líneas vacías seleccionando esta opción. La parte inferior del objeto del list box se deja vacía:

Days	Values
Monday	5
Tuesday	6
Wednesday	41
Thursday	66
Friday	12
Saturday	2
Sunday	88

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
hideExtraBlankRows	booleano	true, false

## Objetos soportados

### List Box

---

## Color de línea

Designa el color de las líneas del objeto. El color puede ser especificado por:

- un nombre de color - como "red"
- un valor HEX - como "# ff0000"
- un valor RVB - como "rgb (255,0,0)"

También puede definir esta propiedad utilizando el comando [OBJECT SET RGB COLORS](#).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
stroke	cadena	un valor css, "transparent", "automatic"

Esta propiedad también está disponible para los objetos basados en texto, en cuyo caso designa tanto el color de la fuente como las líneas del objeto, ver [Color de la fuente](#).

## Objetos soportados

## Ancho de línea

Designa el grosor de una línea.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
strokeWidth	number	0 para el ancho más pequeño en un formulario impreso, o cualquier valor de entero < 20

### Objetos soportados

[Línea](#) - [Óvalo](#) - [Rectángulo](#)

---

## Array colores de fondo

### List box de tipo array

El nombre de un array para aplicar un color de fondo personalizado a cada línea o columna del list box.

Debe introducirse el nombre de un array Entero largo. Cada elemento de este array corresponde a una línea del list box (si se aplica al list box) o a una celda de la columna (si se aplica a una columna), por lo que el array debe tener el mismo tamaño que el array asociado a la columna. Puede utilizar las constantes del tema [SET RGB COLORS](#). Si desea que la celda herede el color de fondo definido en el nivel superior, pase el valor -255 al elemento del array correspondiente.

Por ejemplo, dado un list box en el que las líneas tienen un color gris/gris claro alternado, definido en las propiedades del list box. También se ha definido para el list box un array de color de fondo con el fin de cambiar a naranja claro el color de las líneas en las que al menos un valor es negativo:

```
<>_BgdColors{$i}:=0x00FFD0B0 // orange  
<>_BgdColors{$i}:=-255 // valor por defecto
```

Header1	Header2	Header3
21483	9031	27290
24151	21990	-923
21351	2982	18009
8089	12898	20941
13001	-802	22059
4321	16826	11303
24082	26214	22380
16680	23651	20403
-2678	24818	29896
25639	2691	9687
28794	26941	21486
26083	21092	13476
27928	4092	15441
19987	28211	21191
7996	6300	4089
-1063	13388	23683
13008	7470	19897
5388	26918	13547
28559	27007	8365
28454	22646	13824

A continuación, quiere colorear las celdas con valores negativos en naranja oscuro. Para ello, se define un array de colores de fondo para cada columna, por ejemplo <>\_BgndColor\_1, <>\_BgndColor\_2 y <>\_BgndColor\_3. Los valores de estos arrays tienen prioridad sobre los definidos en las propiedades del list box, así como los del array de color de fondo general:

```
<>_BgndColorsCol_3{2}:=0x00FF8000 // naranja oscuro
<>_BgndColorsCol_2{5}:=0x00FF8000
<>_BgndColorsCol_1{9}:=0x00FF8000
<>_BgndColorsCol_1{16}:=0x00FF8000
```

Header1	Header2	Header3
21483	9031	27290
24151	21990	-923
21351	2982	18009
8089	12898	20941
13001	-802	22059
4321	16826	11303
24082	26214	22380
16680	23651	20403
-2678	24818	29896
25639	2691	9687
28794	26941	21486
26083	21092	13476
27928	4092	15441
19987	28211	21191
7996	6300	4089
-1063	13388	23683
13008	7470	19897
5388	26918	13547
28559	27007	8365
28454	22646	13824

Puede obtener el mismo resultado utilizando los comandos `LISTBOX SET ROW FONT STYLE` y `LISTBOX SET ROW COLOR`. Tienen la ventaja de permitirle omitir el tener que predefinir arrays de estilo/color para las columnas: en su lugar son creadas dinámicamente por los comandos.

Nombre	Tipos de datos	Valores posibles
rowFillSource	cadena	El nombre de un array entero largo.

Objetos soportados

[List Box - Columna List Box](#)

---

## Transparente

Define el fondo del list box como "Transparent". Cuando se define, se ignora cualquier [color de fondo alternativo](#) o [color de fondo](#) definido para la columna.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
fill	texto	"transparent"

Objetos soportados

[List Box](#)

Ver también

[Color de fondo / Color de relleno](#)

# Coordenadas y dimensiones

## Altura de línea automática

Esta propiedad sólo está disponible para los list box de tipo array, no jerárquicos. Esta propiedad no está seleccionada por defecto.

Cuando se utiliza, la altura de cada línea de la columna será calculada automáticamente por 4D, y se tendrá en cuenta el contenido de la columna. Tenga en cuenta que sólo se tendrán en cuenta las columnas con la opción seleccionada para calcular el alto de línea.

Al redimensionar el formulario, si la propiedad de [dimensionamiento horizontal](#) "Agrandar" fue asignada al list box, la columna más a la derecha se agrandará más allá de su ancho máximo, si es necesario.

Cuando esta propiedad está activada, la altura de cada línea se calcula automáticamente para que el contenido de la celda quepa por completo sin ser truncado (a menos que la opción [Wordwrap](#) esté desactivada).

- El cálculo de la altura de línea tiene en cuenta:
  - todo tipo de contenido (texto, números, fechas, horas, imágenes (el cálculo depende del formato de la imagen), objetos),
  - todo tipo de control (entradas, casillas de selección, listas, listas desplegables),
  - fuentes, estilos y tamaños de letra,
  - la opción [Wordwrap](#): si está desactivada, la altura se basa en el número de párrafos (las líneas se truncan); si está activada, la altura se basa en el número de líneas (no se trunca).
- El cálculo de la altura de línea no tiene en cuenta:
  - contenido de columna oculta
  - La propiedad Row Height Array no se tiene en cuenta para los list box jerárquicos.

Algunos objetos pueden tener una altura predefinida que no se puede modificar.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
rowHeightAuto	booleano	true, false

## Objetos soportados

### [Columna de list box](#)

## Abajo

Coordenadas inferiores del objeto en el formulario.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
bottom	number	mínimo: 0

## Objetos soportados

Área 4D View Pro - Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Combo Box - Lista desplegable - Group Box - Lista jerárquica - Área de entrada - List Box - Línea - Columna List Box - Óvalo - Botón imagen - Menú emergente de imagen - Área de plugins - Indicadores de progreso - Botón radio - Rectángulo - Regla/20> - Spinner - Splitter - Imagen estática Stepper - Sub-formulario - Pestaña - Área de texto - Área Web

---

## Izquierda

Coordenadas de izquierda del objeto en el formulario.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
left	number	mínimo: 0

## Objetos soportados

Área 4D View Pro - Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Combo Box - Lista desplegable - Group Box - Lista jerárquica - Área de entrada - List Box - Línea - Columna List Box - Óvalo - Botón imagen - Menú emergente de imagen - Área de plugins - Indicadores de progreso - Botón radio - Regla - Rectángulo - Spinner - Splitter - Imagen estática Stepper - Sub-formulario - Pestaña - Área de texto - Área Web

---

## Derecha

Coordenadas de derecha del objeto en el formulario.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
right	number	mínimo: 0

## Objetos soportados

Área 4D View Pro - Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Combo Box - Lista desplegable - Group Box - Lista jerárquica - Área de entrada - List Box - Línea - Columna List Box - Óvalo - Botón imagen - Menú emergente de imagen - Área de plugins - Indicadores de progreso - Botón radio - Regla - Rectángulo - Spinner - Splitter - Imagen estática Stepper - Sub-formulario - Pestaña - Área de texto - Área Web

---

## Arriba

Coordenadas superiores del objeto en el formulario.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
top	number	mínimo: 0

## Objetos soportados

Área 4D View Pro - Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Combo Box - Lista desplegable - Group Box - Lista jerárquica - Área de entrada - List Box - Línea - Columna List Box - Óvalo - Botón imagen - Menú emergente de imagen - Área de plugins - Indicadores de progreso - Botón radio - Regla - Rectángulo -

## Radio de redondeo

Define la redondez de las esquinas (en píxeles) de los objetos de tipo [rectángulo](#). Por defecto, el valor del radio de los rectángulos es de 0 píxeles. Puede cambiar esta propiedad para dibujar rectángulos redondeados con formas personalizadas:



El valor mínimo es 0, en este caso se dibuja un rectángulo estándar no redondeado. El valor máximo depende del tamaño del rectángulo (no puede superar la mitad del tamaño del lado más corto del rectángulo) y se calcula dinámicamente.

También se puede definir esta propiedad utilizando los comandos [OBJECT Get corner radius](#) y [OBJECT SET CORNER RADIUS](#).

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
borderRadius	integer	mínimo: 0

### Objetos soportados

[Rectángulo](#)

## Altura

Esta propiedad designa el tamaño vertical de un objeto.

Algunos objetos pueden tener una altura predefinida que no se puede modificar.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
height	number	mínimo: 0

### Objetos soportados

[Área 4D View Pro](#) - [Área 4D Write Pro](#) - [Botón](#) - [Rejilla de botones](#) - [Casilla de selección](#) - [Combo Box](#) - [Lista desplegable](#) - [Group Box](#) - [Lista jerárquica](#) - [Área de entrada](#) - [List Box](#) - [Línea](#) - [Columna List Box](#) - [Óvalo](#) - [Botón imagen](#) - [Menú emergente de imagen](#) - [Área de plugins](#) - [Indicadores de progreso](#) - [Botón radio](#) - [Regla](#) - [Rectángulo](#) - [Spinner](#) - [Splitter](#) - [Imagen estática Stepper](#) - [Sub-formulario](#) - [Pestaña](#) - [Área de texto](#) - [Área Web](#)

---

## Ancho

Esta propiedad designa el tamaño horizontal de un objeto.

- Algunos objetos pueden tener una altura predefinida que no se puede modificar.
- Si la propiedad [Redimensionable](#) se utiliza para una [columna de list box](#), el usuario también puede cambiar

manualmente el tamaño de la columna.

- Al redimensionar el formulario, si la propiedad de [dimensionamiento horizontal "Agrandar"](#) fue asignada al list box, la columna más a la derecha se agrandará más allá de su ancho máximo, si es necesario.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
ancho	number	mínimo: 0

### Objetos soportados

[Área 4D View Pro](#) - [Área 4D Write Pro](#) - [Botón](#) - [Rejilla de botones](#) - [Casilla de selección](#) - [Combo Box](#) - [Lista desplegable](#) - [Group Box](#) - [Lista jerárquica](#) - [Área de entrada](#) - [List Box](#) - [Línea](#) - [Columna List Box](#) - [Óvalo](#) - [Botón imagen](#) - [Menú emergente de imagen](#) - [Área de plugins](#) - [Indicadores de progreso](#) - [Botón radio](#) - [Regla](#) - [Rectángulo](#) - [Spinner](#) - [Splitter](#) - [Imagen estática Stepper](#) - [Sub-formulario](#) - [Pestaña](#) - [Área de texto](#) - [Área Web](#)

## Ancho máximo

El ancho máximo de la columna (en píxeles). El ancho de la columna no puede aumentarse más allá de este valor al redimensionar la columna o el formulario.

Al redimensionar el formulario, si la propiedad de [dimensionamiento horizontal "Agrandar"](#) fue asignada al list box, la columna más a la derecha se agrandará más allá de su ancho máximo, si es necesario.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
maxWidth	number	mínimo: 0

### Objetos soportados

[Columna de list box](#)

## Ancho mínimo

El ancho mínimo de la columna (en píxeles). El ancho de la columna no puede reducirse más allá de este valor al redimensionar la columna o el formulario.

Al redimensionar el formulario, si la propiedad de [dimensionamiento horizontal "Agrandar"](#) fue asignada al list box, la columna más a la derecha se agrandará más allá de su ancho máximo, si es necesario.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
minWidth	number	mínimo: 0

### Objetos soportados

[Columna de list box](#)

## Altura de las líneas

Define la altura de las líneas del list box (excluyendo los encabezados y pies de página). Por defecto, la altura de la línea se define según la plataforma y el tamaño de la fuente.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
rowHeight	cadena	valor css en la unidad "em" o "px" (por defecto)

Objetos soportados

[List Box](#)

Ver también

[Array altura de las líneas](#)

## Array altura de las líneas

Esta propiedad se utiliza para indicar el nombre de un array de altura de línea que se quiere asociar al list box. Un array de altura de línea debe ser de tipo numérico (entero largo por defecto).

Cuando se define un array de altura de línea, cada uno de sus elementos cuyo valor es diferente de 0 (cero) se tiene en cuenta para determinar la altura de la línea correspondiente en el list box, basándose en la unidad de altura de línea actual.

Por ejemplo, puede escribir:

```
ARRAY LONGINT(RowHeights;20)
RowHeights{5}:=3
```

Asumiendo que la unidad de las líneas es "líneas", entonces la quinta línea del list box tendrá una altura de tres líneas, mientras que todas las demás líneas mantendrán su altura por defecto.

- Para los list box de tipo array, esta propiedad sólo está disponible si la opción [Altura de línea automática](#) no está seleccionada.
- Al redimensionar el formulario, si la propiedad de [dimensionamiento horizontal "Agrandar"](#) fue asignada al list box, la columna más a la derecha se agrandará más allá de su ancho máximo, si es necesario.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
rowHeightSource	cadena	Nombre de una variable array 4D.

Objetos soportados

[List Box](#)

Ver también

[Altura de las líneas](#)

# Corte

---

## Columnas

Define el número de columnas de una tabla de miniaturas.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
columnCount	integer	mínimo: 1

Objetos soportados

[Botón imagen](#) - [Rejilla de botones](#) - [Menú desplegable imagen](#)

---

## Rows

Define el número de líneas de una tabla de miniaturas.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
rowCount	integer	mínimo: 1

Objetos soportados

[Botón imagen](#) - [Rejilla de botones](#) - [Menú desplegable imagen](#)

# Fuente de datos

## Inserción automática

Cuando se selecciona esta opción, si un usuario introduce un valor que no se encuentra en la lista asociada al objeto, este valor se añade automáticamente a la lista almacenada en memoria.

Cuando la opción inserción automática no está definida (por defecto), el valor introducido se almacena en el objeto formulario pero no en la lista en memoria.

Esta propiedad es soportada por:

- [Combo box](#) and [list box column](#) form objects associated to a choice list.
- [Combo box](#) form objects whose associated list is filled by their array or object datasource.

Por ejemplo, dada una lista de selección que contiene "Francia, Alemania, Italia" que está asociada a un combo box "Países": si la propiedad inserción automática está activada y un usuario introduce "España", entonces el valor "España" se añade automáticamente a la lista en memoria:



If the choice list was created from a list defined in Design mode, the original list is not modified.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
automaticInsertion	booleano	true, false

## Objetos soportados

[Combo Box - Columna List Box](#)

## Lista de selección

Asocia una lista de selección a un objeto. Puede ser un nombre de lista de elección (una referencia de lista) o una colección de valores por defecto.

You can also associate choice lists to objects using the [OBJECT SET LIST BY NAME](#) or [OBJECT SET LIST BY REFERENCE](#) commands.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
choiceList	list, collection	Una lista de valores posibles
lista	list, collection	Una lista de valores posibles (listas jerárquicas únicamente)

## Objetos soportados

## Lista de selección (lista estática)

Lista de valores estáticos a utilizar como etiquetas para el objeto de control de pestañas.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
labels	list, collection	Una lista de valores para llenar el control de pestañas

Objetos soportados

[Pestañas](#)

---

## Elemento actual

`List boxes colección o entity selection`

Especifica una variable o expresión a la que se asignará el elemento/entidad de la colección seleccionado por el usuario. Debe utilizar una variable objeto o una expresión assignable que acepte objetos. Si el usuario no selecciona nada o si ha utilizado una colección de valores escalares, se asigna el valor Null.

Esta propiedad es de "sólo lectura", se actualiza automáticamente según las acciones del usuario en el list box. No se puede editar su valor para modificar el estado de selección del list box.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
currentItemSource	cadena	Expresión del objeto

Objetos soportados

[List Box](#)

---

## Posición elemento actual

`List boxes colección o entity selection`

Indica una variable o expresión a la que se le asignará un entero largo que indica la posición del elemento/entidad de colección seleccionado por el usuario.

- si no se selecciona ningún elemento/entidad, la variable o expresión recibe cero,
- si se selecciona un solo elemento/entidad, la variable o expresión recibe su ubicación,
- si se seleccionan varios elementos/entidades, la variable o expresión recibe la posición del elemento/entidad que se seleccionó de última.

Esta propiedad es de "sólo lectura", se actualiza automáticamente según las acciones del usuario en el list box. No se puede editar su valor para modificar el estado de selección del list box.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
currentItemPositionSource	cadena	Expresión numérica

Objetos soportados

#### List Box

---

## Data Type (expression type)

Defines the data type for the displayed expression. This property is used with:

- [List box columns](#) of the selection and collection types.
- [Drop-down lists](#) associated to objects or arrays.

See also [Expression Type](#) section.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
dataSourceTypeHint	cadena	<ul style="list-style-type: none"> <li>• list box columns: "boolean", "number", "picture", "text", date", "time". <i>Array/selection list box only: "integer", "object"</i></li> <li>• drop-down lists: "object", "arrayText", "arrayDate", "arrayTime", "arrayNumber"</li> </ul>

Objetos soportados

[Drop-down Lists](#) associated to objects or arrays - [List Box column](#)

---

## Tipo de datos (lista)

Defines the type of data to save in the field or variable associated to the [drop-down list](#). This property is used with:

- Drop-down lists [associated to a choice list](#).
- Drop-down lists [associated to a hierarchical choice list](#).

Hay tres opciones disponibles:

- List reference: declares that the drop-down list is hierarchical. It means that the drop-down list can display up to two hierarchical levels and its contents can be managed by the 4D language commands of the Hierarchical Lists theme.
- Selected item value (default): the drop-down list is not hierarchical and the value of the item chosen in the list by the user is saved directly. For example, if the user chooses the value "Blue", then this value is saved in the field.
- Selected item reference: the drop-down list is not hierarchical and the reference of the choice list item is saved in the object. This reference is the numeric value associated with each item either through the `itemRef` parameter of the [APPEND TO LIST](#) or [SET LIST ITEM](#) commands, or in the list editor. This option lets you optimize memory usage: storing numeric values in fields uses less space than storing strings. It also makes it easier to translate applications: you just create multiple lists in different languages but with the same item references, then load the list based on the language of the application.

Using the Selected item reference option requires compliance with the following principles:

- To be able to store the reference, the field or variable data source must be of the Number type (regardless of the type of value displayed in the list). The [expression](#) property is automatically set.
- Valid and unique references must be associated with list items.
- The drop-down list must be associated with a field or a variable.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
saveAs	cadena	"value", "reference"

Setting only `"dataSourceTypeHint" : "integer"` with a `"type": "dropdown"` form object will declare a hierarchical drop-down list.

## Objetos soportados

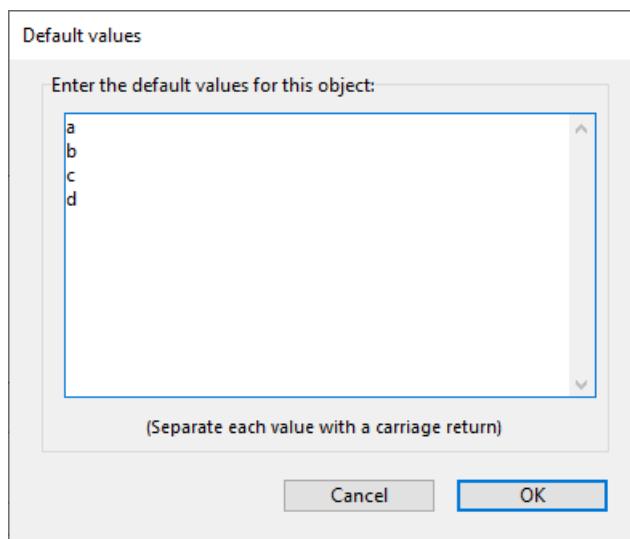
[Drop-down Lists](#) associated to lists

## Valores por defecto (lista de)

List of values that will be used as default values for the list box column (array type only). These values are automatically available in the [array variable](#) associated with this column when the form is executed. Using the language, you can manage the object by referring to this array.

Do not make confusion between this property and the "[default value](#)" property that allows to define a field value in new records.

Debe introducir una lista de valores. In the Form editor, a specific dialog box allows you to enter values separated by carriage returns:



You can also define a [choice list](#) with the list box column. However, a choice list will be used as list of selectable values for each column row, whereas the default list fill all column rows.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
values	colección	A collection of default values (strings), ex: "a", "b", "c", "d"

## Objetos soportados

[Columna List Box \(sólo tipo array\)](#)

# Expresión

This description is specific to [selection](#) and [collection](#) type list box columns. See also [Variable or Expression](#) section.

Una expresión 4D que se asociará a una columna. Puede introducir:

- A simple variable (in this case, it must be explicitly declared for compilation). You can use any type of variable except BLOBs and arrays. The value of the variable will be generally calculated in the [On Display Detail](#) event.
- A field using the standard [Table]Field syntax ([selection type list box](#) only), for example: `[Employees]LastName`. Se pueden utilizar los siguientes tipos de campos:
  - Cadena
  - Numeric
  - Fecha
  - Hora
  - Imagen
  - Boolean

Puede utilizar campos de la tabla maestra o de otras tablas.

- A 4D expression (simple expression, formula or 4D method). La expresión debe devolver un valor. The value will be evaluated in the [On Display Detail](#) and [On Data Change](#) events. The result of the expression will be automatically displayed when you switch to Application mode. The expression will be evaluated for each record of the selection (current or named) of the Master Table (for selection type list boxes), each element of the collection (for collection type list boxes) or each entity of the selection (for entity selection list boxes). If it is empty, the column will not display any results.

The following expression types are supported:

- Cadena
- Numeric
- Fecha
- Imagen
- Booleano

For collection/entity selection list boxes, Null or unsupported types are displayed as empty strings.

When using collections or entity selections, you will usually declare the element property or entity attribute associated to a column within an expression containing `This`. `This` is a dedicated 4D command that returns a reference to the currently processed element. For example, you can use `This.<propertyPath>` where `<propertyPath>` is the path of a property in the collection or an entity attribute path to access the current value of each element/entity.

If you use a collection of scalar values, 4D will create an object for each collection element with a single property (named "value"), filled with the element value. In this case, you will use `This.value` as expression.

If a [non-assignable expression](#) is used (e.g. `[Person]FirstName" "+[Person]LastName`), the column is never enterable even if the [Enterable](#) property is enabled.

If a field, a variable, or an assignable expression (e.g. `Person.lastName`) is used, the column can be enterable or not depending on the [Enterable](#) property.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
dataSource	cadena	Una variable 4D, un nombre de campo o una expresión del lenguaje compleja arbitraria.

## Objetos soportados

### Columna de list box

## Tabla principal

### Current selection list boxes

Specifies the table whose current selection will be used. This table and its current selection will form the reference for the fields associated with the columns of the list box (field references or expressions containing fields). Even if some columns contain fields from other tables, the number of rows displayed will be defined by the master table.

All database tables can be used, regardless of whether the form is related to a table (table form) or not (project form).

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
tabla	number	Número de tabla

### Objetos soportados

#### List Box

---

## Guardar como

This property is available in the following conditions:

- a [choice list](#) is associated with the object
- for [inputs](#) and [list box columns](#), a [required list](#) is also defined for the object (both options should use usually the same list), so that only values from the list can be entered by the user.

This property specifies, in the context of a field or variable associated with a list of values, the type of contents to save:

- Save as Value (default option): the value of the item chosen in the list by the user is saved directly. For example, if the user chooses the value "Blue", then this value is saved in the field.
- Save as Reference: the reference of the choice list item is saved in the object. This reference is the numeric value associated with each item either through the *itemRef* parameter of the [APPEND TO LIST](#) or [SET LIST ITEM](#) commands, or in the list editor.

This option lets you optimize memory usage: storing numeric values in fields uses less space than storing strings. It also makes it easier to translate applications: you just create multiple lists in different languages but with the same item references, then load the list based on the language of the application.

Using this property requires compliance with the following principles:

- To be able to store the reference, the field or variable data source must be of the Number type (regardless of the type of value displayed in the list). The [expression](#) property is automatically set.
- Valid and unique references must be associated with list items.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
saveAs	cadena	"value", "reference"

### Objetos soportados

#### [Input - List Box Column](#)

---

## Elementos seleccionados

### List boxes colección o entity selection

Specifies a variable or expression that will be assigned the elements or entities selected by the user.

- for a collection list box, you must use a collection variable or an assignable expression that accepts collections,
- for an entity selection list box, an entity selection object is built. Debe utilizar una variable objeto o una expresión assignable que acepte objetos.

Esta propiedad es de "sólo lectura", se actualiza automáticamente según las acciones del usuario en el list box. No se puede editar su valor para modificar el estado de selección del list box.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
selectedItemsSource	cadena	Collection expression

## Objetos soportados

### [List Box](#)

---

## Selección de nombres

### [List boxes de tipo selección nombrada](#)

Especifica la selección con nombre que se utilizará. You must enter the name of a valid named selection. It can be a process or interprocess named selection. The contents of the list box will be based on this selection. The named selection chosen must exist and be valid at the time the list box is displayed, otherwise the list box will be displayed blank.

Las selecciones con nombre son listas ordenadas de registros. They are used to keep the order and current record of a selection in memory. For more information, refer to Named Selections section in the *4D Language Reference manual*.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
namedSelection	cadena	Nombre de la selección

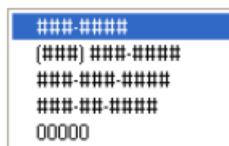
## Objetos soportados

### [List Box](#)

# Visualización

## Formato Alfa

Alpha formats control the way the alphanumeric fields and variables appear when displayed or printed. Here is a list of formats provided for alphanumeric fields:



You can choose a format from this list or use any custom format. The default list contains formats for some of the most common alpha fields that require formats: US telephone numbers (local and long distance), Social Security numbers, and zip codes. You can also enter a custom format name set in the Filters and formats editor of the tool box. In this case, the format cannot be modified in the object properties. Any custom formats or filters that you have created are automatically available, preceded by a vertical bar (|).

The number sign (#) is the placeholder for an alphanumeric display format. You can include the appropriate dashes, hyphens, spaces, and any other punctuation marks that you want to display. You use the actual punctuation marks you want and the number sign for each character you want to display.

For example, consider a part number with a format such as "RB-1762-1".

El formato alfa sería:

```
##-#####-#
```

When the user enters "RB17621," the field displays:

```
RB-1762-1
```

El campo contiene realmente "RB17621".

If the user enters more characters than the format allows, 4D displays the last characters. Por ejemplo, si el formato es:

```
(#####)
```

and the user enters "proportion", the field displays:

```
(portion)
```

El campo contiene realmente "proportion". 4D accepts and stores the entire entry no matter what the display format. No se pierde ninguna información.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
textFormat	cadena	"### #####", "(##) ### ####", "### ### ####", "### ## ####", "00000", formatos personalizados

Objetos soportados

[Drop-down List](#) - [Combo Box](#) - [List Box Column](#) - [List Box Footer](#)

## Formato Fecha

Date formats control the way dates appear when displayed or printed. For data entry, you enter dates in the MM/DD/YYYY format, regardless of the display format you have chosen.

Unlike [Alpha](#) and [Number](#) formats, display formats for dates must only be selected among the 4D built-in formats.

La siguiente tabla muestra las opciones disponibles:

Nombre del formato	Cadena JSON	Ejemplo (sistema USA)
System date short	- (por defecto)	03/25/20
System date abbreviated (1)	systemMedium	Wed, Mar 25, 2020
System date long	systemLong	Wednesday, March 25, 2020
RFC 822	rfc822	Tue, 25 Mar 2020 22:00:00 GMT
Short Century	shortCentury	03/25/20 pero 04/25/2032 (2)
Internal date long	largo	March 25, 2020
Fecha interna abreviada (1)	abbreviated	Mar 25, 2020
Internal date short	short	03/25/2020
ISO Date Time (3)	iso8601	2020-03-25T00:00:00

(1) To avoid ambiguity and in accordance with current practice, the abbreviated date formats display "jun" for June and "jul" for July. This particularity only applies to French versions of 4D.

(2) The year is displayed using two digits when it belongs to the interval (1930;2029) otherwise it will be displayed using four digits. This is by default but it can be modified using the [SET DEFAULT CENTURY](#) command.

(3) The [ISO Date Time](#) format corresponds to the XML date and time representation standard (ISO8601). It is mainly intended to be used when importing/exporting data in XML format and in Web Services.

Regardless of the display format, if the year is entered with two digits then 4D assumes the century to be the 21st if the year belongs to the interval (00;29) and the 20th if it belongs to the interval (30;99). This is the default setting but it can be modified using the [SET DEFAULT CENTURY](#) command.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
dateFormat	cadena	"systemShort", "systemMedium", "systemLong", "iso8601", "rfc822", "short", "shortCentury", "abbreviated", "long", "blankIfNull" (puede combinarse con otros valores posibles)

## Formato de número

Number fields include the Integer, Long integer, Integer 64 bits, Real and Float types.

Number formats control the way numbers appear when displayed or printed. For data entry, you enter only the numbers (including a decimal point or minus sign if necessary), regardless of the display format you have chosen.

4D ofrece varios formatos de números por defecto.

### Marcadores

In each of the number display formats, the number sign (#), zero (0), caret (^), and asterisk (\*) are used as placeholders. You create your own number formats by using one placeholder for each digit you expect to display.

Marcador	Efecto para cero inicial o posterior
#	No muestra nada
0	Muestra 0
^	Muestra un espacio (1)
*	Muestra un asterisco

(1) The caret (^) generates a space character that occupies the same width as a digit in most fonts.

For example, if you want to display three-digit numbers, you could use the format ###. If the user enters more digits than the format allows, 4D displays <<< in the field to indicate that more digits were entered than the number of digits specified in the display format.

If the user enters a negative number, the leftmost character is displayed as a minus sign (unless a negative display format has been specified). If ##0 is the format, minus 26 is displayed as -26 and minus 260 is displayed as <<< because the minus sign occupies a placeholder and there are only three placeholders.

No matter what the display format, 4D accepts and stores the number entered in the field. No se pierde ninguna información.

Each placeholder character has a different effect on the display of leading or trailing zeros. A leading zero is a zero that starts a number before the decimal point; a trailing zero is a zero that ends a number after the decimal point.

Suppose you use the format ##0 to display three digits. If the user enters nothing in the field, the field displays 0. Si el usuario introduce 26, el campo muestra 26.

### Caracteres separadores

The numeric display formats (except for scientific notations) are automatically based on regional system parameters. 4D replaces the “.” and “,” characters by, respectively, the decimal separator and the thousand separator defined in the operating system. The period and comma are thus considered as placeholder characters, following the example of 0 or #.

On Windows, when using the decimal separator key of the numeric keypad, 4D makes a distinction depending on the type of field where the cursor is located: \* in a Real type field, using this key will insert the decimal separator defined in the system, \* in any other type of field, this key inserts the character associated with the key, usually a period (.) or comma (,).

## Puntos decimales y otros caracteres de visualización

You can use a decimal point in a number display format. If you want the decimal to display regardless of whether the user types it in, it must be placed between zeros.

Puede utilizar cualquier otro carácter en el formato. When used alone, or placed before or after placeholders, the characters always appear. Por ejemplo, si utiliza el siguiente formato:

```
$##0
```

a dollar sign always appears because it is placed before the placeholders.

If characters are placed between placeholders, they appear only if digits are displayed on both sides. Por ejemplo, si define el formato:

```
###.##0
```

the point appears only if the user enters at least four digits.

Spaces are treated as characters in number display formats.

## Formatos para positivo, negativo y cero

A number display format can have up to three parts allowing you to specify display formats for positive, negative, and zero values. You specify the three parts by separating them with semicolons as shown below:

```
Positivo;Negativo;Cero
```

You do not have to specify all three parts of the format. If you use just one part, 4D uses it for all numbers, placing a minus sign in front of negative numbers.

If you use two parts, 4D uses the first part for positive numbers and zero and the second part for negative numbers. If you use three parts, the first is for positive numbers, the second for negative numbers, and the third for zero.

The third part (zero) is not interpreted and does not accept replacement characters. If you enter `###;###;#`, the zero value will be displayed "#". In other words, what you actually enter is what will be displayed for the zero value.

Here is an example of a number display format that shows dollar signs and commas, places negative values in parentheses, and does not display zeros:

```
¥###,##0.00; (¥###,##0.00);
```

Notice that the presence of the second semicolon instructs 4D to use nothing to display zero. The following format is similar except that the absence of the second semicolon instructs 4D to use the positive number format for zero:

```
¥###,##0.00; (¥###,##0.00)
```

En este caso, la visualización del cero sería \$0.00.

## Notación científica

If you want to display numbers in scientific notation, use the ampersand (&) followed by a number to specify the number of digits you want to display. Por ejemplo, el formato:

mostrará 759.62 como:

```
7.60e+2
```

The scientific notation format is the only format that will automatically round the displayed number. Note in the example above that the number is rounded up to 7.60e+2 instead of truncating to 7.59e+2.

## Formatos hexadecimales

You can display a number in hexadecimal using the following display formats:

- `&x` : This format displays hexadecimal numbers using the "0xFFFF" format.
- `&$` : este formato muestra números hexadecimales utilizando el formato "\$FFFF".

## Notación XML

The `&xml` format will make a number compliant with XML standard rules. In particular, the decimal separator character will be a period "." in all cases, regardless of the system settings.

## Mostrar un número como una hora

You can display a number as a time (with a time format) by using `&/` followed by a digit. Time is determined by calculating the number of seconds since midnight that the value represents. The digit in the format corresponds to the order in which the time format appears in the Format drop-down menu.

Por ejemplo, el formato:

```
&/5
```

corresponds to the 5th time format in the pop-up menu, specifically the AM/PM time. A number field with this format would display 25000 as:

```
6:56 AM
```

## Ejemplos

The following table shows how different formats affect the display of numbers. The three columns — Positive, Negative, and Zero — each show how 1,234.50, -1,234.50, and 0 would be displayed.

Formato introducido	Positivo	Negativo	Cero
## #	<<<	<<<	
### #	1234	<<<<	
#### ##	1234	-1234	
#### .##	1234.5	-1234.5	
### ##0.00	1234.50	-1234.50	0.00
### ##0	1234	-1234	0
+#### #0;-#### #0;0	+1234	-1234	0
#### #0DB;#### #0CR;0	1234DB	1234CR	0
#### #0;(#### #0)	1234	(1234)	0
###,##0	1,234	-1,234	0
##,##0.00	1,234.50	-1,234.50	0.00
~~~~~^	1234	-1234	
~~~~~^0	1234	-1234	0
~,^0	1,234	-1,234	0
~,^0.00	1,234.50	-1,234.50	0.00
*****	***1234	**-1234	*****
*****0	***1234	**-1234	*****0
**,##0	**1,234	*-1,234	*****0
*,##0.00	*1,234.50	-1,234.50	*****0.00
\$*,##0.00;-\$*,##0.00	\$1,234.50	-\$1,234.50	\$*****0.00
\$~~~~^0	\$ 1234	\$-1234	\$ 0
\$~~~^0;-\$~~~^0	\$1234	-\$1234	\$ 0
\$~~~^0 ;(\$~~~^0)	\$1234	(\$1234)	\$ 0
\$~,^0.00 ;(\$~,^0.00)	\$1,234.50	(\$1,234.50)	\$ 0.00
&2	1.2e+3	-1.2e+3	0.0e+0
&5	1.23450e+3	-1.23450e+3	0.00000
&xml	1234.5	-1234.5	0

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
numberFormat	cadena	Numbers (including a decimal point or minus sign if necessary)

## Objetos soportados

[Combo Box](#) - [Drop-down List](#) - [Input](#) - [List Box Column](#) - [List Box Footer](#) - [Progress Indicators](#)

## Formato imagen

Picture formats control how pictures appear when displayed or printed. For data entry, the user always enters pictures by pasting them from the Clipboard or by drag and drop, regardless of the display format.

The truncation and scaling options do not affect the picture itself. El contenido de un campo Imagen siempre se guarda. Only the display on the particular form is affected by the picture display format.

## A escala para ajustarse

Gramática JSON: "scaled"

The Scaled to fit format causes 4D to resize the picture to fit the dimensions of the area.



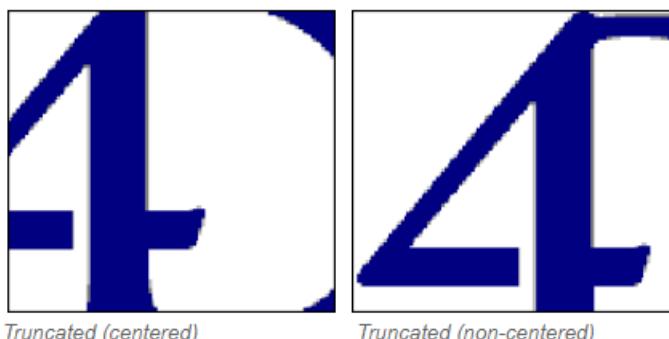
## Truncado (centrado y no centrado)

JSON grammar: "truncatedCenter" / "truncatedTopLeft"

The Truncated (centered) format causes 4D to center the picture in the area and crop any portion that does not fit within the area. 4D crops equally from each edge and from the top and bottom.

The Truncated (non-centered) format causes 4D to place the upper-left corner of the picture in the upper-left corner of the area and crop any portion that does not fit within the area. 4D crops from the right and bottom.

When the picture format is Truncated (non-centered), it is possible to add scroll bars to the input area.



## Scaled to fit (proportional) and Scaled to fit centered (proportional)

JSON grammar: "proportionalTopLeft" / "proportionalCenter"

When you use Scaled to fit (proportional), the picture is reduced proportionally on all sides to fit the area created for the picture. The Scaled to fit centered (proportional) option does the same, but centers the picture in the picture area.

If the picture is smaller than the area set in the form, it will not be modified. If the picture is bigger than the area set in the form, it is proportionally reduced. Since it is proportionally reduced, the picture will not appear distorted.

If you have applied the Scaled to fit centered (proportional) format, the picture is also centered in the area:



Scaled to fit (proportional)



Scaled to fit centered (proportional)

## Replicado

Gramática JSON: "tiled"

When the area that contains a picture with the Replicated format is enlarged, the picture is not deformed but is replicated as many times as necessary in order to fill the area entirely.



If the field is reduced to a size smaller than that of the original picture, the picture is truncated (non-centered).

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
pictureFormat	cadena	"truncatedTopLeft", "scaled", "truncatedCenter", "tiled", "proportionalTopLeft", "proportionalCenter"

### Objetos soportados

[Input - List Box Column - List Box Footer](#)

---

## Formato Hora

Time formats control the way times appear when displayed or printed. For data entry, you enter times in the 24-hour HH: MM:SS format or the 12-hour HH: MM:SS AM/PM format, regardless of the display format you have chosen.

Unlike [Alpha](#) and [Number](#) formats, display formats for times must only be selected among the 4D built-in formats.

The table below shows the Time field display formats and gives examples:

Nombre del formato	Cadena JSON	Comentarios	Ejemplo para 04:30:25
HH:MM:SS	hh_mm_ss		04:30:25
HH:MM	hh_mm		04:30
Hour Min Sec	HH_MM_SS		4 horas 30 minutos 25 segundos
Hour Min	HH_MM		4 horas 30 minutos
HH:MM AM/PM	hh_mm_am		4:30 a.m.
MM SS	mm_ss	Hora expresada como duración a partir de las 00:00:00	270:25
Min Sec	MM_SS	Hora expresada como duración a partir de las 00:00:00	270 Minutos 25 Segundos
ISO Date Time	iso8601	Corresponds to the XML standard for representing time-related data. It is mainly intended to be used when importing/exporting data in XML format	0000-00-00T04:30:25
System time short	- (por defecto)	Formato de hora estándar definido en el sistema	04:30:25
System time long abbreviated	systemMedium	macOS only: Abbreviated time format defined in the system. Windows: this format is the same as the System time short format	4•30•25 AM
System time long	systemLong	macOS only: Long time format defined in the system. Windows: this format is the same as the System time short format	4:30:25 AM HNEC

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
timeFormat	cadena	"systemShort", "systemMedium", "systemLong", "iso8601", "hh_mm_ss", "hh_mm", "hh_mm_am", "mm_ss", "HH_MM_SS", "HH_MM", "MM_SS", "blankIfNull" (puede combinarse con otros valores posibles)

## Objetos soportados

[Combo Box](#) - [Drop-down List](#) - [Input](#) - [List Box Column](#) - [List Box Footer](#)

---

## Texto cuando False/Texto cuando True

When a [boolean expression](#) is displayed as:

- a text in an [input object](#)
- a "[popup](#)" in a [list box column](#),

... you can select the text to display for each value:

- Text when True - the text to be displayed when the value is "true"
- Text when False - the text to be displayed when the value is "false"

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
booleanFormat	cadena	"<textWhenTrue>;<textWhenFalse>", e.g. "Assigned;Unassigned"

Objetos soportados

[List Box Column - Input](#)

---

## Tipo de visualización

Used to associate a display format with the column data. The formats provided depends on the variable type (array type list box) or the data/field type (selection and collection type list boxes).

Boolean and number (numeric or integer) columns can be displayed as check boxes. In this case, the [Title](#) property can be defined.

Boolean columns can also be displayed as pop-up menus. In this case, the [Text when False](#) and [Text when True](#) properties must be defined.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
controlType	cadena	<ul style="list-style-type: none"> <li>number columns: "automatic" (default) or "checkbox"</li> <li>boolean columns: "checkbox" (default) or "popup"</li> </ul>

Objetos soportados

[Columna de list box](#)

---

## No renderizado

When this property is enabled, the object is not drawn on the form, however it can still be activated.

In particular, this property allows implementing "invisible" buttons. Non-rendered buttons can be placed on top of graphic objects. They remain invisible and do not highlight when clicked, however their action is triggered when they are clicked.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
display	booleano	true, false

Objetos soportados

[Button - Drop-down List](#)

---

## Tres estados

Permite que un objeto casilla de selección acepte un tercer estado. La variable asociada a la casilla de selección devuelve el valor 2 cuando la casilla está en el tercer estado.

Casillas de verificación de tres estados en columnas list box

List box columns with a numeric [data type](#) can be displayed as three-states check boxes. Si se elige, se muestran los

siguientes valores:

- 0 = casilla no seleccionada,
- 1 = casilla seleccionada,
- 2 (or any value >0) = semi-checked box (third state). Para la entrada de datos, este estado devuelve el valor 2.
- -1 = casilla de verificación invisible,
- -2 = unchecked box, not enterable,
- -3 = checked box, not enterable,
- -4 = casilla semi-marcada, no editable

In this case as well, the [Title](#) property is also available so that the title of the check box can be entered.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
threeState	booleano	true, false

Objetos soportados

[Check box - List Box Column](#)

---

## Título

This property is available for a list box column if:

- the [column type](#) is boolean and its [display type](#) is "Check Box"
- the [column type](#) is number (numeric or integer) and its [display type](#) is "Three-states Checkbox".

In that cases, the title of the check box can be entered using this property.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
controlTitle	cadena	Any custom label for the check box

Objetos soportados

[Columna de list box](#)

---

## Truncar con puntos suspensivos

Controls the display of values when list box columns are too narrow to show their full contents.

This option is available for columns with any type of contents, except pictures and objects.

- When the property is enabled (default), if the contents of a list box cell exceed the width of the column, they are truncated and an ellipsis is displayed:

Cities	Popu...
Charlotte	792862
New York	8406...
Philadelp...	1553...
San Fran...	837442
San Jose	288054

La posición de la elipsis depende del sistema operativo. In the above example (Windows), it is added on the right

side of the text. On macOS, the ellipsis is added in the middle of the text.

- When the property is disabled, if the contents of a cell exceed the width of the column, they are simply clipped with no ellipsis added:

Cities	Popu...
Charlotte	792862
New York	3406000
Philadelphia	1553000
San Francisco	837442
San Jose	288054

The Truncate with ellipsis option is enabled by default and can be specified with list boxes of the Array, Selection, or Collection type.

When applied to Text type columns, the Truncate with ellipsis option is available only if the [Wordwrap](#) option is not selected. When the Wordwrap property is selected, extra contents in cells are handled through the word-wrapping features so the Truncate with ellipsis property is not available.

The Truncate with ellipsis property can be applied to Boolean type columns; however, the result differs depending on the [cell format](#):

- For Pop-up type Boolean formats, labels are truncated with an ellipsis,
- For Check box type Boolean formats, labels are always clipped.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
truncateMode	cadena	"withEllipsis", "none"

## Objetos soportados

[List Box Column](#) - [List Box Header](#)

---

## Visibilidad

This property allows hiding the object in the Application environment.

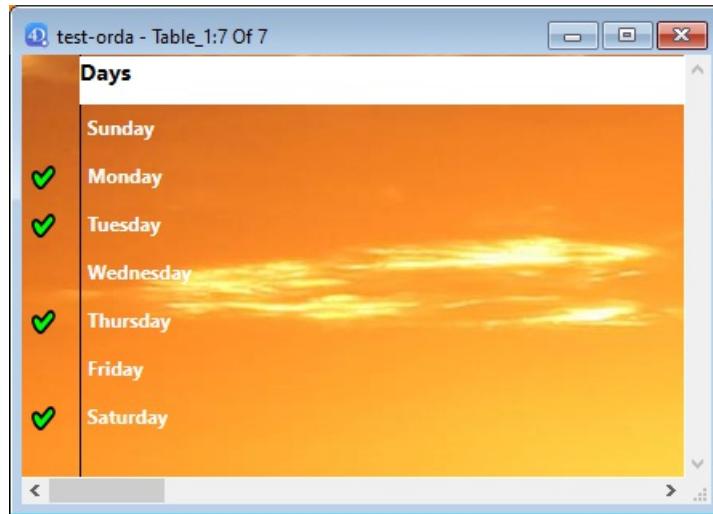
You can handle the Visibility property for most form objects. This property is mainly used to simplify dynamic interface development. In this context, it is often necessary to hide objects programatically during the `On load` event of the form then to display certain objects afterwards. In this context, it is often necessary to hide objects programatically during the `On load` event of the form then to display certain objects afterwards. The Visibility property allows inverting this logic by making certain objects invisible by default.

### Visibilidad automática en los formularios lista

In the context of ["list" forms](#), the Visibility property supports two specific values:

- If record selected (JSON name: "selectedRows")
- If record not selected (JSON name: "unselectedRows")

This property is only used when drawing objects located in the body of a list form. It tells 4D whether or not to draw the object depending on whether the record being processed is selected/not selected. It allows you to represent a selection of records using visual attributes other than highlight colors:



4D does not take this property into account if the object was hidden using the `OBJECT SET VISIBLE` command; in this case, the object remains invisible regardless of whether or not the record is selected.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
visibility	cadena	"visible", "hidden", "selectedRows" (list form only), "unselectedRows" (list form only)

#### Objetos soportados

4D View Pro area - 4D Write Pro area - Button - Button Grid - Check Box - Combo Box - Drop-down List - Group Box - Hierarchical List - List Box - List Box Column - List Box Footer - List Box Header - Picture Button - Picture Pop-up Menu - Plug-in Area - Progress indicator - Radio Button - Spinner - Splitter - Static Picture - Stepper - Subform - Tab control - Text Area - Web Area

## Ajuste de texto

For `input` objects, available when the `Multiline` property is set to "yes" .

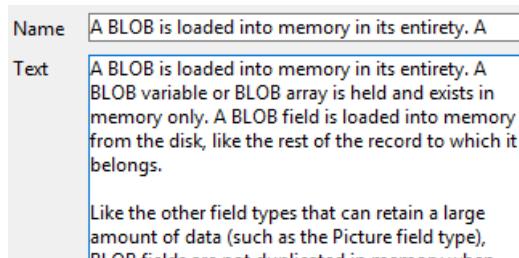
Gestiona la visualización del contenido cuando supera el ancho del objeto.

Checked for list box/Yes for input

Gramática JSON: "normal"

When this option is selected, text automatically wraps to the next line whenever its width exceeds that of the column/area, if the column/area height permits it.

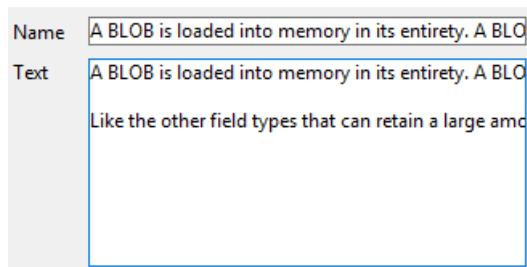
- In single-line columns/areas, only the last word that can be displayed entirely is displayed. 4D inserts line returns; it is possible to scroll the contents of the area by pressing the down arrow key.
- In multiline columns/areas, 4D carries out automatic line returns.



Unchecked for list box/No for input

#### Gramática JSON: "none"

When this option is selected, 4D does not do any automatic line returns and the last word that can be displayed may be truncated. In text type areas, carriage returns are supported:



In list boxes, any text that is too long is truncated and displayed with an ellipse (...). In the following example, the Wordwrap option is checked for the left column and unchecked for the right column:

Header1	Header2
The vertical alignment will be applied as long as the full text fits in the cell. Otherwise, the text will be aligned to the top so as the beginning of the text will visible.	The vertical alignment will be ap...
You can make a field invisible in the Application environment and for the plug-ins by selecting the Invisible property for this field.	You can make a field invisible in ...

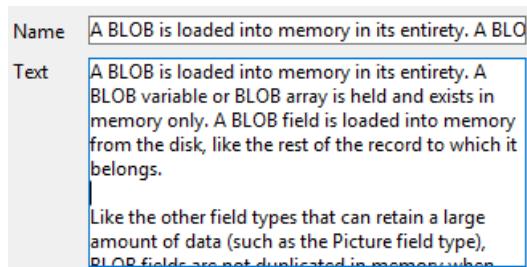
Note that regardless of the Wordwrap option's value, the row height is not changed. If the text with line breaks cannot be entirely displayed in the column, it is truncated (without an ellipse). In the case of list boxes displaying just a single row, only the first line of text is displayed:

Header1	Header2
The vertical alignment will be	The vertical alignment will be ap...
You can make a field invisible in	You can make a field invisible in ...

#### Automático para entrada (opción por defecto)

##### Gramática JSON: "automatic"

- In single-line areas, words located at the end of lines are truncated and there are no line returns.
- In multiline areas, 4D carries out automatic line returns.



#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
wordwrap	cadena	"automatic" (excluyendo list box), "normal", "none"

#### Objetos soportados

[Input - List Box Column - List Box Footer](#)

# Entrada

---

## Corrección ortográfica automática

4D incluye funcionalidades de corrección ortográfica integradas y personalizables. Text type [inputs](#) can be checked, as well as [4D Write Pro](#) documents.

The Auto Spellcheck property activates the spell-check for each object. When used, a spell-check is automatically performed during data entry. You can also execute the `SPELL CHECKING` 4D language command for each object to be checked.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
spellcheck	booleano	true, false

### Objetos soportados

[Área 4D Write Pro](#) - [Área de entrada](#)

---

## Menú contextual

Allows the user access to a standard context menu in the object when the form is executed.

For a picture type [input](#), in addition to standard editing commands (Cut, Copy, Paste and Clear), the menu contains the Import... command, which can be used to import a picture stored in a file, as well as the Save as... command, which can be used to save the picture to disk. The menu can also be used to modify the display format of the picture: the Truncated non-centered, Scaled to fit and Scaled to fit centered prop. options are provided. The modification of the [display format](#) using this menu is temporary; it is not saved with the record.

For a [multi-style](#) text type [input](#), in addition to standard editing commands, the context menu provides the following commands:

- Fonts...: displays the font system dialog box
- Recent fonts: displays the names of recent fonts selected during the session. The list can store up to 10 fonts (beyond that, the last font used replaces the oldest). By default, this list is empty and the option is not displayed. You can manage this list using the `SET RECENT FONTS` and `FONT LIST` commands.
- commands for supported style modifications: font, size, style, color and background color. When the user modifies a style attribute via this pop-up menu, 4D generates the `On After Edit` form event.

For a [Web Area](#), the contents of the menu depend of the rendering engine of the platform. It is possible to control access to the context menu via the `WA SET PREFERENCE` command.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
contextMenu	cadena	"automatic" (se utiliza si falta), "none"

### Objetos soportados

[Input](#) - [Web Area](#) - [4D Write Pro areas](#)

---

## Editable

The Enterable attribute indicates whether users can enter values into the object.

Los objetos son editables por defecto. If you want to make a field or an object non-enterable for that form, you can disable the Enterable property for the object. Un objeto no editable sólo muestra datos. You control the data by methods that use the field or variable name. You can still use the `On Clicked`, `On Double Clicked`, `On Drag Over`, `On Drop`, `On Getting Focus` and `On Losing Focus` form events with non-enterable objects. This makes it easier to manage custom context menus and lets you design interfaces where you can drag-and-drop and select non-enterable variables.

When this property is disabled, any pop-up menus associated with a list box column via a list are disabled.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
editable	booleano	true, false

### Objetos soportados

[4D Write Pro areas](#) - [Check Box](#) - [Hierarchical List](#) - [Input](#) - [List Box Column](#) - [Progress Bar](#) - [Ruler](#) - [Stepper](#)

---

## Filtro de entrada

An entry filter controls exactly what the user can type during data entry. Unlike [required lists](#) for example, entry filters operate on a character-by-character basis. For example, if a part number always consists of two letters followed by three digits, you can use an entry filter to restrict the user to that pattern. You can even control the particular letters and numbers.

Un filtro de entrada sólo funciona durante la entrada de datos. It has no effect on data display after the user deselects the object. In general, you use entry filters and [display formats](#) together. The filter constrains data entry and the format ensures proper display of the value after data entry.

During data entry, an entry filter evaluates each character as it is typed. If the user attempts to type an invalid character (a number instead of a letter, for example), 4D simply does not accept it. The null character remains unchanged until the user types a valid character.

Entry filters can also be used to display required formatting characters so that the user need not enter them. For example, an American telephone number consists of a three-digit area code, followed by a seven-digit number that is broken up into two groups of three and four digits, respectively. A display format can be used to enclose the area code in parentheses and display a dash after the third digit of the telephone number. When such a format is used, the user does not need to enter the parentheses or the dashes.

### Definir un filtro de entrada

Most of the time, you can use one of the [built-in filters](#) of 4D for what you need; however, you can also create and use custom filters:

- puede introducir directamente una cadena de definición de filtro
- or you can enter the name of an entry filter created in the Filters editor in the Toolbox. The names of custom filters you create begin with a vertical bar (!).

For information about creating entry filters, see [Filter and format codes](#).

### Filtros de entrada por defecto

Here is a table that explains each of the entry filter choices in the Entry Filter drop-down list:

Filtro de entrada	Descripción
~A	Permitir la entrada de toda letra, pero convertir a mayúsculas.
&9	Permitir sólo números.
&A	Permitir sólo letras mayúsculas.
&a	Permitir sólo letras (mayúsculas y minúsculas).
&@	Permitir sólo caracteres alfanuméricos. No hay caracteres especiales.
~a##	Abreviatura del nombre del estado (por ej., CA). Permite la entrada de dos letras, pero las convierte en mayúsculas.
!0&9##/#/#/#	Formato de entrada de fechas estándar. Mostrar ceros en los espacios de entrada. Permitir cualquier número.
!0&9 Día: ## Mes: ## Año: ##	Formato de entrada de hora. Mostrar ceros en los espacios de entrada. Permitir cualquier número. Limitado a horas y minutos.
!0&9##:##	Formato de entrada de hora. Limitado a horas y minutos. Mostrar ceros en los espacios de entrada. Permitir cuatro números, separados por dos puntos.
!0&9## Horas ## Minutos ## Segundos	Formato de entrada de hora. Mostrar ceros en los espacios de entrada. Permitir dos números antes de cada palabra.
!0&9Horas: ## Minutas: ## Segundos: ##	Formato de entrada de hora. Mostrar ceros en los espacios de entrada. Permitir dos números después de cada palabra.
!0&9##-##-##-##	Formato de número de teléfono local. Mostrar ceros en los espacios de entrada. Permitir cualquier número. Tres entradas, guión, cuatro entradas.
!_&9(###)0###- ###	Número de teléfono de larga distancia. Display underscores in first three entry spaces, zeros in remainder.
!0&9##-##-##-##	Número de teléfono de larga distancia. Mostrar ceros en los espacios de entrada. Permitir cualquier número. Three entries, hyphen, three entries, hyphen, four entries.
!0&9##-##-##	Número de la Seguridad Social. Mostrar ceros en los espacios de entrada. Permitir cualquier número.
~"A-Z;0-9; ;.;;-"	Uppercase letters and punctuation. Allow only capital letters, numbers, spaces, commas, periods, and hyphens.
&"a-z;0-9; ;.;;-"	Letras mayúsculas y minúsculas y puntuación. Allow lowercase letters, numbers, spaces, commas, periods, and hyphens.
&"0-9;.;;-"	Números. Allow only numbers, decimal points, and hyphens (minus sign).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
entryFilter	cadena	<ul style="list-style-type: none"> <li>Código de filtro de entrada o</li> <li>Entry filter name (filter names start with   )</li> </ul>

## Objetos soportados

[Check Box](#) - [Combo Box](#) - [Hierarchical List](#) - [Input](#) - [List Box Column](#)

## Focusable

When the Focusable property is enabled for an object, the object can have the focus (and can thus be activated by the keyboard for instance). It is outlined by a gray dotted line when it is selected — except when the [Hide focus rectangle](#)

option has also been selected.

An [input object](#) is always focusable if it has the [Enterable](#) property.

- **Current Member**

Casilla de verificación muestra el foco cuando se selecciona

- **Current Member**

La casilla de verificación está seleccionada pero no se puede mostrar el foco|

When the Focusable property is selected for a non-enterable object, the user can select, copy or even drag-and-drop the contents of the area.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
focusable	booleano	true, false

#### Objetos soportados

[4D Write Pro areas](#) - [Button](#) - [Check Box](#) - [Drop-down List](#) - [Hierarchical List](#) - [Input](#) - [List Box](#) - [Plug-in Area](#) - [Radio Button](#) - [Subform](#)

## Disposición del teclado

This property associates a specific keyboard layout to an [input object](#). For example, in an international application, if a form contains a field whose contents must be entered in Greek characters, you can associate the "Greek" keyboard layout with this field. This way, during data entry, the keyboard configuration is automatically changed when this field has the focus.

By default, the object uses the current keyboard layout.

You can also set and get the keyboard dynamically using the `OBJECT SET KEYBOARD LAYOUT` and `OBJECT Get keyboard layout` commands.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
keyboardDialect	texto	Language code, for example "ar-ma" or "cs". Ver RFC3066, ISO639 e ISO3166

#### Objetos soportados

[4D Write Pro areas](#) - [Input](#)

## Multilínea

Esta propiedad está disponible para [objetos de entrada](#) que contienen expresiones de tipo Texto y campos de tipo Alfa y Texto. Puede tener tres valores diferentes: Sí, No, Automático (por defecto).

#### Automático

- En las entradas de una línea, las palabras situadas al final de las líneas se truncan y no hay retornos de línea.

- En las entradas multilínea, 4D realiza retornos de línea automáticos:

Name	A BLOB is loaded into memory in its entirety. A BLO
Text	A BLOB is loaded into memory in its entirety. A BLOB variable or BLOB array is held and exists in memory only. A BLOB field is loaded into memory from the disk, like the rest of the record to which it belongs.  Like the other field types that can retain a large amount of data (such as the Picture field type), <a href="#">BLOB fields are not duplicated in memory when</a>

No

- En las entradas de una línea, las palabras situadas al final de las líneas se truncan y no hay retornos de línea.
- Nunca hay retornos de línea: el texto siempre se muestra en una sola línea. Si el campo o la variable Alfa o Texto contiene retornos de carro, el texto situado después del primer retorno de carro se elimina en cuanto se modifica el área:

Name	A BLOB is loaded into memory in its entirety. A BLO
Text	A BLOB is loaded into memory in its entirety. A BLO

Sí

Cuando se selecciona este valor, la propiedad es gestionada por la opción [Retorno de línea](#).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
multilínea	texto	"yes", "no", "automatic" (por defecto si no se define)

## Objetos soportados

### Entrada

---

## Marcador

4D can display placeholder text in the fields of your forms.

Placeholder text appears as watermark text in a field, supplying a help tip, indication or example for the data to be entered. This text disappears as soon as the user enters a character in the area:

Product Number	Enter 12-digit product code
Product Number	1

The placeholder text is displayed again if the contents of the field is erased.

A placeholder can be displayed for the following types of data:

- cadena (text o alpha)
- date and time when the Blank if null property is enabled.

You can use an XLIFF reference in the ":xliiff:resname" form as a placeholder, for example:

```
:xliiff:PH_Lastname
```

You only pass the reference in the "Placeholder" field; it is not possible to combine a reference with static text.

You can also set and get the placeholder text by programming using the [OBJECT SET PLACEHOLDER](#) and [OBJECT Get placeholder](#) commands.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
placeholder	cadena	Texto a mostrar (en gris) cuando el objeto no contiene ningún valor

Objetos soportados

[Combo Box - Área de entrada](#)

Ver también

[Mensaje de ayuda](#)

## Selección siempre visible

This property keeps the selection visible within the object after it has lost the focus. This makes it easier to implement interfaces that allow the text style to be modified (see [Multi-style](#)).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
showSelection	booleano	true, false

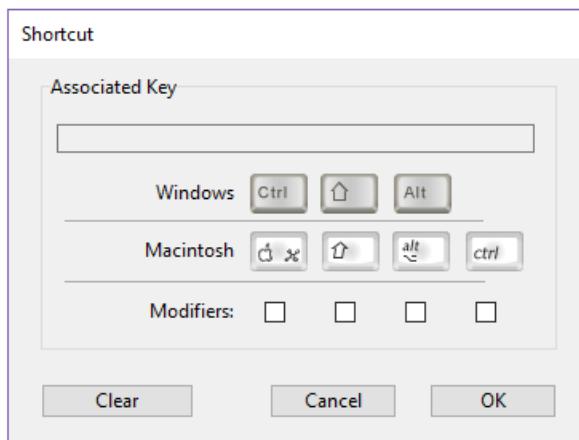
Objetos soportados

[4D Write Pro areas - Input](#)

## Atajo

This property allows setting special meaning keys (keyboard shortcuts) for [buttons](#), [radio buttons](#), and [checkboxes](#). They allow the user to use the control using the keyboard instead of having to use the mouse.

You can configure this option by clicking the [...] button in the Shortcuts property in the Property List.



You can also assign a shortcut to a custom menu command. If there is a conflict between two shortcuts, the

active object has priority. For more information about associating shortcuts with menus, refer to [Setting menu properties](#).

To view a list of all the shortcuts used in the 4D Design environment, see the [Shortcuts Page](#) in the Preferences dialog box.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
shortcutAccel	booleano	true, false (Windows: Ctrl/macOS: Command)
shortcutAlt	booleano	true, false
shortcutCommand	booleano	true, false
shortcutControl	booleano	true, false (macOS: Control)
shortcutShift	booleano	true, false
shortcutKey	cadena	<ul style="list-style-type: none"><li>• toda llave de carácter: "a", "b"...</li><li>• [F1]" -&gt; "[F15]", "[Return]", "[Enter]", "[Backspace]", "[Tab]", "[Esc]", "[Del]", "[Home]", "[End]", "[Help]", "[Page up]", "[Page down]", "[left arrow]", "[right arrow]", "[up arrow]", "[down arrow]"</li></ul>

## Objetos soportados

[Button](#) - [Check Box](#) - [Picture Button](#) - [Radio Button](#)

## Edición con un solo clic

Permite el paso directo al modo de edición en list boxes.

When this option is enabled, list box cells switch to edit mode after a single user click, regardless of whether or not this area of the list box was selected beforehand. Note that this option allows cells to be edited even when the list box [selection mode](#) is set to "None".

When this option is not enabled, users must first select the cell row and then click on a cell in order to edit its contents.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
singleClickEdit	booleano	true, false

## Objetos soportados

[List Box](#)

# Pies

---

## Mostrar pies

Esta propiedad se utiliza para mostrar u ocultar [los pies de columna listbox](#). Hay un pie de página por columna; cada pie de página se configura por separado.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
showFooters	booleano	true, false

### Objetos soportados

#### List Box

---

## Altura

Esta propiedad se utiliza para definir la altura de línea de un pie de list box en píxeles o líneas de texto (cuando se muestra). Ambos tipos de unidades pueden utilizarse en el mismo list box:

- *Píxel* - el valor de la altura se aplica directamente a la línea en cuestión, independientemente del tamaño de la fuente contenida en las columnas. Si una fuente es demasiado grande, el texto se trunca. Además, las imágenes se truncan o cambian de tamaño según su formato.
- *Línea* - la altura se calcula teniendo en cuenta el tamaño de la fuente de la línea en cuestión.
  - Si se define más de un tamaño, 4D utiliza el mayor. Por ejemplo, si una línea contiene "Verdana 18", "Geneva 12" y "Arial 9", 4D utiliza "Verdana 18" para determinar la altura de la línea (por ejemplo, 25 píxeles). Esta altura se multiplica por el número de líneas definidas.
  - Este cálculo no tiene en cuenta el tamaño de las imágenes ni los estilos aplicados a las fuentes.
  - En macOS, la altura de línea puede ser incorrecta si el usuario introduce caracteres que no están disponibles en la fuente seleccionada. Cuando esto ocurre, se utiliza un tipo de letra sustituto, lo que puede provocar variaciones en el tamaño.

This property can also be set dynamically using the [LISTBOX SET FOOTERS HEIGHT](#) command.

Conversión de unidades: cuando se pasa de una unidad a otra, 4D las convierte automáticamente y muestra el resultado en la Lista de propiedades. Por ejemplo, si la fuente utilizada es "Lucida grande 24", una altura de "1 línea" se convierte en "30 píxeles" y una altura de "60 píxeles" se convierte en "2 líneas".

Tenga en cuenta que la conversión de ida y vuelta puede conducir a un resultado final diferente del valor inicial debido a los cálculos automáticos realizados por 4D. Esto se ilustra en las siguientes secuencias:

(font Arial 18): 52 pixels -> 2 lines -> 40 pixels (font Arial 12): 3 pixels -> 0.4 line rounded up to 1 line -> 19 pixels

Ejemplo JSON:

```
"List Box": {  
    "type": "listbox",  
    "showFooters": true,  
    "footerHeight": "44px",  
    ...  
}
```

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
footerHeight	cadena	decimales positivos +px   em

## Objetos soportados

[List Box](#)

Ver también

[Encabezados - Pies List box](#)

# Rejillas

---

## Color líneas horizontales

Define el color de las líneas horizontales de un list box (gris por defecto).

Gramática JSON

Nombre	Tipos de datos	Valores posibles
horizontalLineStroke	color	todo valor css, "transparent", "automatic"

Objetos soportados

[List Box](#)

---

## Color líneas verticales

Define el color de las líneas verticales de un list box (gris por defecto).

Gramática JSON

Nombre	Tipos de datos	Valores posibles
verticalLineStroke	color	todo valor css, "transparent", "automatic"

Objetos soportados

[List Box](#)

# Encabezados

---

## Mostrar encabezados

Esta propiedad se utiliza para mostrar u ocultar [los encabezados de columna listbox](#). Hay un encabezado por columna; cada encabezado se configura por separado.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
showHeaders	booleano	true, false

### Objetos soportados

#### List Box

---

## Altura

Esta propiedad se utiliza para definir la altura de línea de un encabezado de list box en píxeles o líneas de texto (cuando se muestra). Ambos tipos de unidades pueden utilizarse en el mismo list box:

- *Píxel* - el valor de la altura se aplica directamente a la línea en cuestión, independientemente del tamaño de la fuente contenida en las columnas. Si una fuente es demasiado grande, el texto se trunca. Además, las imágenes se truncan o cambian de tamaño según su formato.
- *Línea* - la altura se calcula teniendo en cuenta el tamaño de la fuente de la línea en cuestión.
  - Si se define más de un tamaño, 4D utiliza el mayor. Por ejemplo, si una línea contiene "Verdana 18", "Geneva 12" y "Arial 9", 4D utiliza "Verdana 18" para determinar la altura de la línea (por ejemplo, 25 píxeles). Esta altura se multiplica por el número de líneas definidas.
  - Este cálculo no tiene en cuenta el tamaño de las imágenes ni los estilos aplicados a las fuentes.
  - En macOS, la altura de línea puede ser incorrecta si el usuario introduce caracteres que no están disponibles en la fuente seleccionada. Cuando esto ocurre, se utiliza un tipo de letra sustituto, lo que puede provocar variaciones en el tamaño.

This property can also be set dynamically using the [LISTBOX SET HEADERS HEIGHT](#) command.

Conversión de unidades: cuando se pasa de una unidad a otra, 4D las convierte automáticamente y muestra el resultado en la Lista de propiedades. Por ejemplo, si la fuente utilizada es "Lucida grande 24", una altura de "1 línea" se convierte en "30 píxeles" y una altura de "60 píxeles" se convierte en "2 líneas".

Tenga en cuenta que la conversión de ida y vuelta puede conducir a un resultado final diferente del valor inicial debido a los cálculos automáticos realizados por 4D. Esto se ilustra en las siguientes secuencias:

(font Arial 18): 52 pixels -> 2 lines -> 40 pixels (font Arial 12): 3 pixels -> 0.4 line rounded up to 1 line -> 19 pixels

Ejemplo JSON:

```
"List Box": {  
    "type": "listbox",  
    "showHeaders": true,  
    "headerHeight": "22px",  
    ...  
}
```

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
headerHeight	cadena	decimales positivos +px   em )

## Objetos soportados

[List Box](#)

Ver también

[Pies - Encabezados List box](#)

# Ayuda

## Mensaje de ayuda

Esta propiedad permite asociar los mensajes de ayuda a los objetos activos de sus formularios. Se pueden mostrar en ejecución:

The screenshot shows a user interface with two input fields. The first field is labeled "Phone :" and contains a placeholder icon. The second field is labeled "Comments :" and contains the text "Enter phone number including the 3-digit area code". A tooltip is visible over the "Comments :" field.

- El retardo de la visualización y la duración máxima de los mensajes de ayuda pueden controlarse utilizando los selectores `Tips delay` y `Tips duration` del comando **SET DATABASE PARAMETER**.
- Los mensajes de ayuda se pueden deshabilitar o habilitar globalmente para la aplicación utilizando el selector del comando **SET DATABASE PARAMETER**.

Puede:

- designar un mensajes de ayuda existente, previamente especificado en el editor de [mensajes de ayuda](#) de 4D.
- o introducir el mensaje de ayuda directamente como una cadena. Esto le permite aprovechar la arquitectura XLIFF. Aquí puede introducir una referencia XLIFF para mostrar un mensaje en el lenguaje de la aplicación (para más información sobre XLIFF, consulte [Anexo B: Arquitectura XLIFF](#)). También puede utilizar referencias 4D (ver [Uso de referencias en texto estático](#)).

En macOS, no se soporta la visualización de mensajes de ayuda en ventanas de tipo emergente.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
tooltip	texto	información adicional para ayudar al usuario

## Objetos soportados

[Botón](#) - [Rejilla de botones](#) - [Casilla de selección](#) - [Lista desplegable](#) - [Combo Box](#) - [Lista jerárquica](#) - [Encabezado List Box](#) - [Pie de List Box](#) - [Botón imagen](#) - [Menú emergente imagen](#) - [Botón Radio](#)

## Otras funcionalidades de ayuda

También puede asociar los mensajes de ayuda a los objetos formulario de otras dos maneras:

- a nivel de la estructura de la base de datos (sólo campos). En este caso, la ayuda del campo se muestra en todos los formularios en los que aparece. Para más información, consulte "Consejos de ayuda" en [Propiedades de los campos](#).
- utilizando el comando **OBJECT SET HELP TIP**, para el proceso actual.

Cuando se asocian consejos diferentes a un mismo objeto en varias ubicaciones, se aplica el siguiente orden de prioridad:

1. nivel de estructura (prioridad más baja)
2. editor de formulario
3. Comando **OBJECT SET HELP TIP** (alta prioridad)

## Ver también

Marcador

# Jerarquía

---

## List box jerárquico

### List box de tipo array

This property specifies that the list box must be displayed in hierarchical form. In the JSON form, this feature is triggered [when the `dataSource` property value is an array](#), i.e. a collection.

Additional options (Variable 1...10) are available when the *Hierarchical List Box* option is selected, corresponding to each `dataSource` array to use as break column. Each time a value is entered in a field, a new row is added. Se pueden especificar hasta 10 variables. These variables set the hierarchical levels to be displayed in the first column.

Ver [List box jerárquicos](#)

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
datasource	array cadena	Colección de nombres de arrays que definen la jerarquía

### Objetos soportados

[List Box](#)

# List Box

---

## Columnas

Colección de columnas del list box.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
columns	colección de objetos columna	Contiene las propiedades de las columnas de list box

Para ver una lista de las propiedades que soportan los objetos columna, consulte la sección [Propiedades específicas de la columna](#).

### Objetos soportados

#### [List Box](#)

---

## Nombre formulario detallado

### `List box del tipo selección`

Especifica el formulario que se utilizará para modificar o mostrar los registros individuales del list box.

Se muestra el formulario especificado:

- when using `Add Subrecord` and `Edit Subrecord` standard actions applied to the list box (see [Using standard actions](#)),
- when a row is double-clicked and the `Double-click on Row` property is set to "Edit Record" or "Display Record".

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
detailForm	cadena	<ul style="list-style-type: none"><li>• Nombre (cadena) de la tabla o formulario proyecto</li><li>• POSIX path (string) to a .json file describing the form</li><li>• Objeto que describe el formulario</li></ul>

### Objetos soportados

#### [List Box](#)

---

## Doble clic en línea

### `List box del tipo selección`

Define la acción a realizar cuando un usuario haga doble clic en una línea en el list box. Las opciones disponibles son:

- No hacer nada (por defecto): hacer doble clic en una línea no desencadena ninguna acción automática.
- Editar registro: al hacer doble clic en una línea se muestra el registro correspondiente en el formulario detallado definido [para el list box](#). El registro se abre en modo de lectura-escritura para que pueda ser modificado.

- Mostrar registro: idéntica a la acción anterior, salvo que el registro se abre en modo de sólo lectura para que no pueda ser modificado.

 Double-clicking an empty row is ignored in list boxes.

Independientemente de la acción seleccionada/elegida, se genera el evento de formulario `On Double clicked`.

Para las dos últimas acciones, también se genera el evento de formulario `On Open Detail`. `On Close Detail` se genera cuando un registro mostrado en el formulario detallado asociado al list box está a punto de cerrarse (independientemente de que el registro se haya modificado o no).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
doubleClickInRowAction	cadena	"editSubrecord", "displaySubrecord"

## Objetos soportados

### List Box

---

## Conjunto resaltado

### List box del tipo selección

This property is used to specify the set to be used to manage highlighted records in the list box (when the `Arrays` data source is selected, a Boolean array with the same name as the list box is used).

4D creates a default set named `ListBoxSetN` where *N* starts at 0 and is incremented according to the number of list boxes in the form. Si es necesario, puede modificar el conjunto por defecto. It can be a local, process or interprocess set (we recommend using a local set, for example `$LBSet`, in order to limit network traffic). A continuación, 4D lo mantiene automáticamente. If the user selects one or more rows in the list box, the set is updated immediately. If you want to select one or more rows by programming, you can apply the commands of the “Sets” theme to this set.

- The highlighted status of the list box rows and the highlighted status of the table records are completely independent.
- If the “Highlight Set” property does not contain a name, it will not be possible to make selections in the list box.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
highlightSet	cadena	Nombre del conjunto

## Objetos soportados

### List Box

---

## Columnas bloqueadas y columnas estáticas

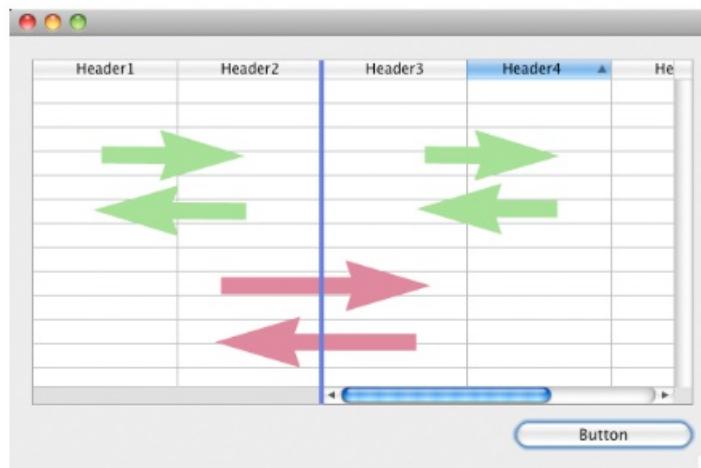
Locked columns and static columns are two separate and independent functionalities in list boxes:

- Locked columns always stay displayed to the left of the list box; they do not scroll horizontally.
- Static columns cannot be moved by drag and drop within the list box.

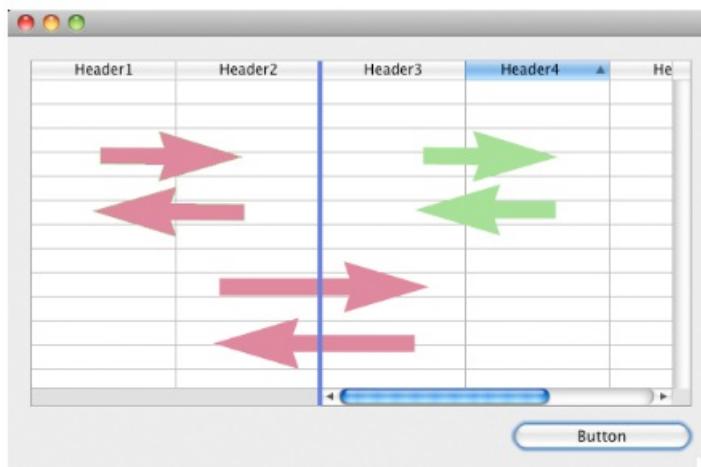
Puede añadir o eliminar columnas de forma dinámica por programación, utilizando comandos como [LISTBOX INSERT COLUMN](#) o [LISTBOX DELETE COLUMN](#).

Estas propiedades interactúan de la siguiente manera:

- If you set columns that are only static, they cannot be moved.
- If you set columns that are locked but not static, you can still change their position freely within the locked area. However, a locked column cannot be moved outside of this locked area.



- If you set all of the columns in the locked area as static, you cannot move these columns within the locked area.



- You can set a combination of locked and static columns according to your needs. For example, if you set three locked columns and one static column, the user can swap the two right-most columns within the locked area (since only the first column is static).

## Número de columnas bloqueadas

Number of columns that must stay permanently displayed in the left part of the list box, even when the user scrolls through the columns horizontally.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
lockedColumnCount	integer	mínimo: 0

## Objetos soportados

### List Box

## Número de columnas estáticas

Número de columnas que no se pueden mover durante la ejecución.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
staticColumnCount	integer	mínimo: 0

## Objetos soportados

### List Box

---

## Número de columnas

Define el número de columnas del list box.

Puede añadir o eliminar columnas de forma dinámica por programación, utilizando comandos como [LISTBOX INSERT COLUMN](#) o [LISTBOX DELETE COLUMN](#).

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
columnCount	integer	mínimo: 1

## Objetos soportados

### List Box

---

## Array de control de líneas

### List box de tipo array

A 4D array controlling the display of list box rows.

You can set the "hidden", "disabled" and "selectable" interface properties for each row in an array-based list box using this array. It can also be designated using the [LISTBOX SET ARRAY](#) command.

The row control array must be of the Longint type and include the same number of rows as the list box. Each element of the *Row Control Array* defines the interface status of its corresponding row in the list box. Three interface properties are available using constants in the "List Box" constant theme:

Constante	Valor	Comentario
lk row is disabled	2	The corresponding row is disabled. The text and controls such as check boxes are dimmed or grayed out. Enterable text input areas are no longer enterable. Valor por defecto: Activado
lk row is hidden	1	La línea correspondiente está oculta. Hiding rows only affects the display of the list box. The hidden rows are still present in the arrays and can be managed by programming. The language commands, more particularly LISTBOX Get number of rows or LISTBOX GET CELL POSITION , do not take the displayed/hidden status of rows into account. For example, in a list box with 10 rows where the first 9 rows are hidden, LISTBOX Get number of rows returns 10. From the user's point of view, the presence of hidden rows in a list box is not visibly discernible. Only visible rows can be selected (for example using the Select All command). Valor por defecto: Visible
lk row is not selectable	4	The corresponding row is not selectable (highlighting is not possible). Enterable text input areas are no longer enterable unless the Single-Click Edit option is enabled. Controls such as check boxes and lists are still functional however. This setting is ignored if the list box selection mode is "None". Valor por defecto: Seleccionable

To change the status for a row, you just need to set the appropriate constant(s) to the corresponding array element. For example, if you do not want row #10 to be selectable, you can write:

```
aLControlArr{10}:=lk row is not selectable
```

RowNum	Countries	Population	Landlocked
1	Luxembourg	0 502 202	<input checked="" type="checkbox"/>
2	Latvia	1 973 700	<input type="checkbox"/>
3	Kuwait	4 044 500	<input type="checkbox"/>
4	Croatia	4 284 889	<input type="checkbox"/>
5	Denmark	5 699 220	<input type="checkbox"/>
6	Nicaragua	6 071 045	<input type="checkbox"/>
7	Czech Republic	10 674 947	<input checked="" type="checkbox"/>
8	Serbia	7 306 677	<input checked="" type="checkbox"/>
9	Honduras	8 249 574	<input type="checkbox"/>
10	Austria	8 572 895	<input checked="" type="checkbox"/>
11	Hungary	10 005 000	<input checked="" type="checkbox"/>
12	Greece	10 815 197	<input type="checkbox"/>
13	Benin	10 879 829	<input type="checkbox"/>

You can define several interface properties at once:

```
aLControlArr{8}:=lk row is not selectable + lk row is disabled
```

RowNum	Countries	Population	Landlocked
1	Luxembourg	0 502 202	<input checked="" type="checkbox"/>
2	Latvia	1 973 700	<input type="checkbox"/>
3	Kuwait	4 044 500	<input type="checkbox"/>
4	Croatia	4 284 889	<input type="checkbox"/>
5	Denmark	5 699 220	<input type="checkbox"/>
6	Nicaragua	6 071 045	<input type="checkbox"/>
7	Czech Republic	10 674 947	<input checked="" type="checkbox"/>
8	Serbia	7 306 677	<input checked="" type="checkbox"/>
9	Honduras	8 249 574	<input type="checkbox"/>
10	Austria	8 572 895	<input checked="" type="checkbox"/>
11	Hungary	10 005 000	<input checked="" type="checkbox"/>
12	Greece	10 815 197	<input type="checkbox"/>
13	Benin	10 879 829	<input type="checkbox"/>

Note that setting properties for an element overrides any other values for this element (if not reset). Por ejemplo:

```

aLControlArr{6}:=lk row is disabled + lk row is not selectable
//sets row 6 as disabled AND not selectable
aLControlArr{6}:=lk row is disabled
//sets row 6 as disabled but selectable again

```

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
rowControlSource	cadena	Nombre del array de control de líneas

## Objetos soportados

[List Box](#)

---

## Modo de selección

Designa la opción para permitir a los usuarios seleccionar líneas:

- Ninguna: las líneas no se pueden seleccionar si se elige este modo. Hacer clic en la lista no tendrá ningún efecto a menos que la opción [Edición con un solo clic](#) esté activada. Las teclas de navegación sólo hacen que la lista se desplace; no se genera el evento de formulario `On Selection Change`.
- Simple: en este modo se puede seleccionar una línea a la vez. Si hace clic en una línea, la seleccionará. Un `Ctrl+clic` (Windows) o `Comando+clic` (macOS) en una línea cambia su estado (entre seleccionada o no). Las teclas de flecha arriba y abajo seleccionan la línea anterior/siguiente de la lista. Las otras teclas de navegación se desplazan por la lista. El evento de formulario `On Selection Change` se genera cada vez que se cambia la línea actual.
- Múltiple: en este modo se pueden seleccionar varias líneas simultáneamente.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
selectionMode	cadena	"multiple", "single", "none"

## Objetos soportados

[List Box](#)

# Objetos

---

## Tipo

### PARÁMETRO OBLIGATORIO

Esta propiedad designa el tipo del [objeto formulario activo o inactivo](#).

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
type	cadena	"button", "buttonGrid", "checkbox", "combo", "dropdown", "groupBox", "input", "line", "list", "listbox", "oval", "picture", "pictureButton", "picturePopup", "plugin", "progress", "radio", "rectangle", "ruler", "spinner", "splitter", "stepper", "subform", "tab", "text", "view", "webArea", "write"

### Objetos soportados

4D View Pro area - 4D Write Pro area - Button - Button Grid - Check Box - Combo Box - Drop-down List - Group Box - Hierarchical List - List Box - List Box Column - List Box Footer - List Box Header - Picture Button - Picture Pop-up Menu - Plug-in Area - Progress indicator - Radio Button - Spinner - Splitter - Static Picture - Stepper - Subform - Tab control - Text Area - Web Area

---

## Nombre del objeto

Each active form object is associated with an object name. Cada nombre de objeto debe ser único.

Los nombres de objetos están limitados a un tamaño de 255 bytes.

When using 4D's language, you can refer to an active form object by its object name (for more information about this, refer to [Object Properties](#) in the 4D Language Reference manual).

For more information about naming rules for form objects, refer to [Identifiers](#) section.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
name	cadena	Any allowed name which does not belong to an already existing object

### Objetos soportados

4D View Pro area - 4D Write Pro area - Button - Button Grid - Check Box - Combo Box - Drop-down List - Group Box - Hierarchical List - List Box - List Box Column - List Box Footer - List Box Header - Picture Button - Picture Pop-up Menu - Plug-in Area - Progress indicator - Spinner - Splitter - Static Picture - Stepper - Radio Button - Subform - Tab control - Text Area - Web Area

---

## Guardar valor

This property is available when the [Save Geometry](#) option is checked for the form.

This feature is only supported for objects that contribute to the overall geometry of the form. For example, this option is available for check boxes because their value can be used to hide or display additional areas in the window.

Here is the list of objects whose value can be saved:

Objeto	Valor guardado
Casilla a seleccionar	Valor de la variable asociada (0, 1, 2)
Lista desplegable	Número de línea seleccionada
Botón radio	Valor de la variable asociada (1, 0, True o False para los botones de acuerdo a su tipo)
Control de pestañas	Número de pestaña seleccionada

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
memorizeValue	booleano	true, false

## Objetos soportados

[Casilla de selección](#) - [Lista desplegable](#) - [Botón de radio](#) - [Control de pestañas](#)

## Variable o expresión

See also [Expression](#) for Selection and collection type list box columns.

Esta propiedad especifica la fuente de los datos. Each active form object is associated with an object name and a variable name. The variable name can be different from the object's name. In the same form, you can use the same variable several times while each [object name](#) must be unique.

El tamaño del nombre de la variable está limitado a 31 bytes. See [Identifiers](#) section for more information about naming rules.

The form object variables allow you to control and monitor the objects. For example, when a button is clicked, its variable is set to 1; at all other times, it is 0. The expression associated with a progress indicator lets you read and change the current setting.

Variables or expressions can be enterable or non-enterable and can receive data of the Text, Integer, Numeric, Date, Time, Picture, Boolean, or Object type.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
dataSource	cadena o array de cadenas	<ul style="list-style-type: none"><li>• Variable, nombre de campo o cualquier expresión 4D.</li><li>• Empty string for <a href="#">dynamic variables</a>.</li><li>• String array (collection of array names) for a <a href="#">hierarchical listbox column</a>]</li></ul>

## Expresiones

You can use an [expression](#) as data source for an object. Any valid 4D expression is allowed: simple expression, object property, formula, 4D function, project method name or field using the standard `[Table]Field` syntax. The expression is evaluated when the form is executed and reevaluated for each form event. Note that expressions can be [assignable](#)

or non-assignable.

If the value entered corresponds to both a variable name and a method name, 4D considers that you are indicating the method.

## Variables dinámicas

You can leave it up to 4D to create variables associated with your form objects (buttons, enterable variables, check boxes, etc.) dynamically and according to your needs. To do this, simply leave the "Variable or Expression" property (or `dataSource JSON` field) blank.

When a variable is not named, when the form is loaded, 4D creates a new variable for the object, with a calculated name that is unique in the space of the process variables of the interpreter (which means that this mechanism can be used even in compiled mode). This temporary variable will be destroyed when the form is closed. In order for this principle to work in compiled mode, it is imperative that dynamic variables are explicitly typed. Hay dos maneras de hacer esto:

- You can set the type using the [Expression type](#) property.
- You can use a specific initialization code when the form is loaded that uses, for example, the `VARIABLE TO VARIABLE` command:

```
If(Form event=On Load)
  C_TEXT($init)
  $Ptr_object:=OBJECT Get pointer(0bject named;"comments")
  $init:=""
  VARIABLE TO VARIABLE(Current process;$Ptr_object->:$init)
End if
```

In the 4D code, dynamic variables can be accessed using a pointer obtained with the `OBJECT Get pointer` command. Por ejemplo:

```
// assign the time 12:00:00 to the variable for the "tstart" object
$p :=OBJECT Get pointer(0bject named;"tstart")
$p->:=?12:00:00?
```

Este mecanismo tiene dos ventajas:

- On the one hand, it allows the development of "subform" type components that can be used several times in the same host form. Let us take as an example the case of a datepicker subform that is inserted twice in a host form to set a start date and an end date. This subform will use objects for choosing the date of the month and the year. It will be necessary for these objects to work with different variables for the start date and the end date. Letting 4D create their variable with a unique name is a way of resolving this difficulty.
- Por otra parte, puede utilizarse para limitar el uso de la memoria. In fact, form objects only work with process or inter-process variables. However, in compiled mode, an instance of each process variable is created in all the processes, including the server processes. This instance takes up memory, even when the form is not used during the session. Therefore, letting 4D create variables dynamically when loading the forms can save memory.

## List box array

For an array list box, the Variable or Expression property usually holds the name of the array variable defined for the list box, and for each column. However, you can use a string array (containing arrays names) as `dataSource` value for a list box column to define a [hierarchical list box](#).

### Objetos soportados

[4D View Pro area](#) - [4D Write Pro area](#) - [Button](#) - [Button Grid](#) - [Check Box](#) - [Combo Box](#) - [Drop-down List](#) - [Hierarchical List](#) - [List Box](#) - [List Box Column](#) - [List Box Header](#) - [List Box Footer](#) - [Picture Pop-up Menu](#) - [Plug-in Area](#) - [Progress indicator](#) - [Spinner](#) - [Splitter](#) - [Stepper](#) - [Tab control](#) - [Subform](#) - [Radio Button](#) - [Web Area](#)

## Tipo de expresión

This property is called **Data Type** in the Property List for **selection** and **collection** type list box columns and for **Drop-down Lists** associated to an **object** or an **array**.

Specify the data type for the expression or variable associated to the object. Note that main purpose of this setting is to configure options (such as display formats) available for the data type. En realidad, no escribe la variable en sí. In view of project compilation, you must [declare the variable](#).

However, this property has a typing function in the following specific cases:

- **Dynamic variables**: you can use this property to declare the type of dynamic variables.
- **List Box Columns**: this property is used to associate a display format with the column data. The formats provided will depend on the variable type (array type list box) or the data/field type (selection and collection type list boxes). The standard 4D formats that can be used are: Alpha, Numeric, Date, Time, Picture and Boolean. The Text type does not have specific display formats. Todos los formatos personalizados existentes también están disponibles.
- **Picture variables**: you can use this menu to declare the variables before loading the form in interpreted mode. Specific native mechanisms govern the display of picture variables in forms. These mechanisms require greater precision when configuring variables: from now on, they must have already been declared before loading the form — i.e., even before the `On Load` form event — unlike other types of variables. To do this, you need either for the statement `C_PICTURE(varName)` to have been executed before loading the form (typically, in the method calling the `DIALOG` command), or for the variable to have been typed at the form level using the expression type property. Otherwise, the picture variable will not be displayed correctly (only in interpreted mode).

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
dataSourceTypeHint	cadena	<ul style="list-style-type: none"><li>• standard objects: "integer", "boolean", "number", "picture", "text", date", "time", "arrayText", "arrayDate", "arrayTime", "arrayNumber", "collection", "object", "undefined"</li><li>• list box columns: "boolean", "number", "picture", "text", date", "time". <i>Array/selection list box only:</i> "integer", "object"</li></ul>

### Objetos soportados

[Check Box](#) - [Combo Box](#) - [Drop-down List](#) - [Input](#) - [List Box Column](#) - [List Box Footer](#) - [Plug-in Area](#) - [Progress indicator](#) - [Radio Button](#) - [Ruler](#) - [Spinner](#) - [Stepper](#) - [Subform](#) - [Tab Control](#)

## Clase CSS

A list of space-separated words used as class selectors in [css files](#).

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
class	cadena	One string with CSS name(s) separated by space characters

### Objetos soportados

[4D View Pro area](#) - [4D Write Pro area](#) - [Button](#) - [Button Grid](#) - [Check Box](#) - [Combo Box](#) - [Drop-down List](#) - [Group Box](#) - [Hierarchical List](#) - [List Box](#) - [Picture Button](#) - [Picture Pop-up Menu](#) - [Plug-in Area](#) - [Radio Button](#) - [Static Picture](#) - [Subform](#) - [Text Area](#) - [Web Area](#)

## Collection o entity selection

To use collection elements or entities to define the row contents of the list box.

Enter an expression that returns either a collection or an entity selection. Usually, you will enter the name of a variable, a collection element or a property that contain a collection or an entity selection.

The collection or the entity selection must be available to the form when it is loaded. Each element of the collection or each entity of the entity selection will be associated to a list box row and will be available as an object through the [This](#) command:

- if you used a collection of objects, you can call This in the datasource expression to access each property value, for example This.<propertyPath>.
- if you used an entity selection, you can call This in the datasource expression to access each attribute value, for example This.<attributePath>.

If you used a collection of scalar values (and not objects), 4D allows you to display each value by calling This.value in the datasource expression. However in this case you will not be able to modify values or to access the current item object (see below)  
Note: For information about entity selections, please refer to the [ORDA](#) chapter.

### Gramática JSON

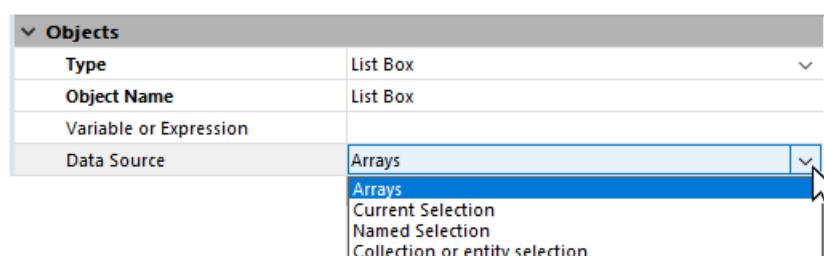
Nombre	Tipos de datos	Valores posibles
dataSource	cadena	Expression that returns a collection or an entity selection.

### Objetos soportados

#### List Box

## Fuente de datos

Especifique el tipo de list box.



- Arrays(default): use array elements as the rows of the list box.
- Current Selection: use expressions, fields or methods whose values will be evaluated for each record of the current selection of a table.
- Named Selection: use expressions, fields or methods whose values will be evaluated for each record of a named selection.
- Collection or Entity Selection: use collection elements or entities to define the row contents of the list box. Note that with this list box type, you need to define the [Collection or Entity Selection](#) property.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
listboxType	cadena	"array", "currentSelection", "namedSelection", "collection"

Objetos soportados

[List Box](#)

---

## Tipo de plug-in

Name of the [plug-in external area](#) associated to the object. Plug-in external area names are published in the manifest.json file of the plug-in.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
pluginAreaKind	cadena	Name of the plug-in external area (starts with a % character)

Objetos soportados

[Área de plug-in](#)

---

## Radio Group

Enables radio buttons to be used in coordinated sets: only one button at a time can be selected in the set.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
radioGroup	cadena	Nombre del grupo radio

Objetos soportados

[Botón radio](#)

---

## Título

Permite insertar una etiqueta en un objeto. The font and the style of this label can be specified.

You can force a carriage return in the label by using the ¥ character (backslash).



Para insertar un ¥ en la etiqueta, ingrese "¥¥".

By default, the label is placed in the center of the object. When the object also contains an icon, you can modify the relative location of these two elements using the [Title/Picture Position](#) property.

For application translation purposes, you can enter an XLIFF reference in the title area of a button (see [Appendix B: XLIFF architecture](#)).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
texto	cadena	todo texto

Objetos soportados

[Button](#) - [Check Box](#) - [List Box Header](#) - [Radio Button](#) - [Text Area](#)

## Variable Calculation

This property sets the type of calculation to be done in a [column footer](#) area.

The calculation for footers can also be set using the [LISTBOX SET FOOTER CALCULATION](#) 4D command.

Hay varios tipos de cálculos disponibles. The following table shows which calculations can be used according to the type of data found in each column and indicates the type automatically affected by 4D to the footer variable (if it is not typed by the code):

Cálculo	Num	Texto	Fecha	Hora	Bool	Imágenes	tipos de variables de pie de página
Mínimo	X	X	X	X	X		Igual que el tipo de columna
Máximo	X	X	X	X	X		Igual que el tipo de columna
Suma	X			X	X		Igual que el tipo de columna
Conteo	X	X	X	X	X	X	Entero largo
Promedio	X			X			Real
Desviación estándar(*)	X			X			Real
Varianza(*)	X			X			Real
Suma de cuadrados(*)	X			X			Real
Personalizado ("None")	X	X	X	X	X	X	Cualquiera

(\*) Sólo para list boxes de tipo array.

Only declared or dynamic [variables](#) can be used to display footer calculations. Other kinds of [expressions](#) such as [Form.value](#) are not supported.

Automatic calculations ignore the shown/hidden state of list box rows. If you want to restrict a calculation to only visible rows, you must use a custom calculation.

*Null* no se tienen en cuenta para ningún cálculo.

If the column contains different types of values (collection-based column for example):

- Average and Sum only take numerical elements into account (other element types are ignored).
- Minimum and Maximum return a result according to the usual type list order as defined in the [collection.sort\(\)](#) function.

Using automatic calculations in footers of columns based upon expressions has the following limitations:

- it is supported with all list box types when the expression is "simple" (such as `[table]field` or `this.attribute`),
- it is supported but not recommended for performance reasons with collection/entity selection list boxes when the expression is "complex" (other than `this.attribute`) and the list box contains a large number of rows,
- it is not supported with current selection/named selection list boxes when the expression is "complex". Es necesario utilizar cálculos personalizados.

When Custom ("none" in JSON) is set, no automatic calculations are performed by 4D and you must assign the value of the variable in this area by programming.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
variableCalculation	cadena	"none", "minimum", "maximum", "sum", "count", "average", "standardDeviation", "variance", "sumSquare"

## Objetos soportados

[Pie de list box](#)

# Imagen

---

## Ruta de acceso

Ruta de una imagen source estática para un [botón imagen](#), [menú emergente de imagen](#), o [imagen estática](#). Debe utilizar la sintaxis POSIX.

The following locations can be used for static pictures:

- in the Resources folder of the project. Appropriate when you want to share static pictures between several forms in the project. En este caso, el nombre de la ruta es "/RESOURCES/<picture path>".
- en una carpeta de imágenes (por ejemplo, llamada Images) dentro de la carpeta del formulario. Apropriado cuando las imágenes estáticas se utilizan sólo en el formulario y/o se quiere poder mover o duplicar todo el formulario dentro del proyecto o de diferentes proyectos. En este caso, el nombre de la ruta es ""<picture path>" y se resuelve desde la raíz de la carpeta del formulario.
- en una variable imagen 4D. The picture must be loaded in memory when the form is executed. In this case, the Pathname is "var:<variableName>".

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
imagen	texto	Relative or filesystem path in POSIX syntax, or "var:<variableName>" for picture variable

## Objetos soportados

[Botón imagen](#) - [Menú emergente imagen](#) - [Imagen estática](#)

---

## Visualización

### A escala para ajustarse

Gramática JSON: "scaled"

The Scaled to fit format causes 4D to resize the picture to fit the dimensions of the area.

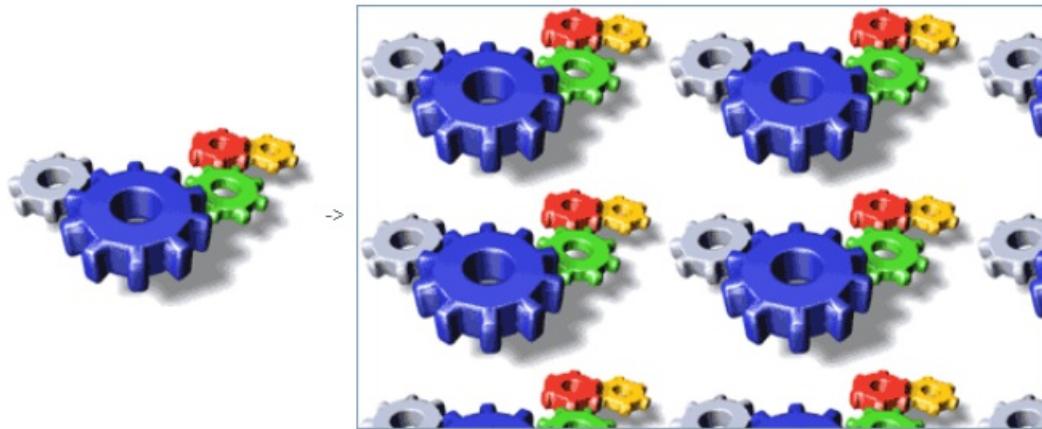


### Replicado

Gramática JSON: "tiled"

When the area that contains a picture with the Replicated format is enlarged, the picture is not deformed but is

replicated as many times as necessary in order to fill the area entirely.



If the field is reduced to a size smaller than that of the original picture, the picture is truncated (non-centered).

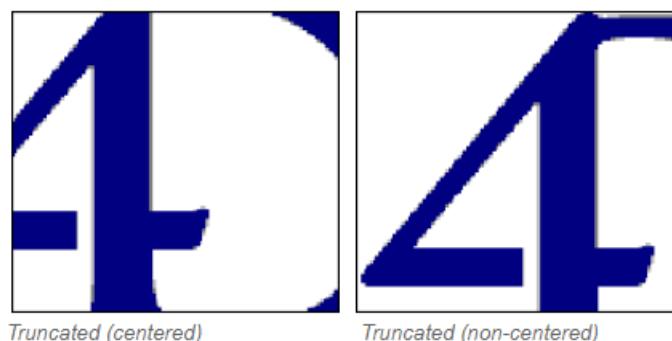
### Centrado / Truncado (no centrado)

JSON grammar: "truncatedCenter" / "truncatedTopLeft"

El formato Centro hace que 4D centre la imagen en el área y recorte cualquier parte que no quiera dentro del área. 4D crops equally from each edge and from the top and bottom.

The Truncated (non-centered) format causes 4D to place the upper-left corner of the picture in the upper-left corner of the area and crop any portion that does not fit within the area. 4D crops from the right and bottom.

When the picture format is Truncated (non-centered), it is possible to add scroll bars to the input area.



### Gramática JSON

Nombre	Tipos de datos	Valores posibles
pictureFormat	cadena	"scaled", "tiled", "truncatedCenter", "truncatedTopLeft"

### Objetos soportados

[Imagen estática](#)

# Plug-ins

---

## Propiedades avanzadas

Si las opciones avanzadas son proporcionadas por el autor del plug-in, un botón Propiedades avanzadas puede ser activado en la lista de propiedades. En este caso, puede hacer clic en este botón para definir estas opciones, normalmente a través de una caja de diálogo personalizada.

Dado que la función de propiedades avanzadas está bajo el control del autor del plug-in, la información sobre estas opciones avanzadas es responsabilidad del distribuidor del plug-in.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
customProperties	texto	Propiedades específicas del plugin, pasadas al plugin como una cadena JSON si es un objeto, o como un buffer binario si es una cadena codificada en base64

### Objetos soportados

### [Área de plug-in](#)

# Imprimir

## Impresión marco

Esta propiedad gestiona el modo de impresión de los objetos cuyo tamaño puede variar de un registro a otro en función de su contenido. Estos objetos pueden configurarse para imprimirse con un marco fijo o variable. Los objetos de marco fijo se imprimen dentro de los límites del objeto tal y como fue creado en el formulario. Los objetos de marco variable se expanden durante la impresión para incluir todo el contenido del objeto. Tenga en cuenta que el ancho de los objetos impresos como tamaño variable no se ve afectado por esta propiedad; sólo la altura varía automáticamente en función del contenido del objeto.

No se puede colocar más de un objeto de marco variable uno al lado del otro en un formulario. Puede colocar objetos de marco no variable a ambos lados de un objeto que se imprimirá con un tamaño variable siempre que el objeto de marco variable sea al menos una línea más largo que el objeto de al lado y que todos los objetos estén alineados en la parte superior. Si no se respeta esta condición, el contenido de los otros campos se repetirá para cada corte horizontal del objeto marco variable.

Los comandos `objeto Print` y `Print form` no soportan esta propiedad.

Las opciones de impresión son:

- La opción Variable / Imprimir marco variable marcada: 4D amplía o reduce el área del objeto del formulario para imprimir todos los subregistros.
- Opción fija (truncamiento) / Imprimir marco variable no seleccionada: 4D sólo imprime el contenido que aparece en el área del objeto. El formulario sólo se imprime una vez y el contenido no impreso se ignora.
- Fijo (Múltiples Registros) (sólo subformularios): se mantiene el tamaño inicial del área del subformulario pero 4D imprime el formulario varias veces para imprimir todos los registros.

Esta propiedad puede definirse por programación utilizando el comando `OBJECT SET PRINT VARIABLE FRAME`.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
printFrame	cadena	"fixed", "variable", (subformulario únicamente) "fixedMultiple"

## Objetos soportados

[Entrada - Subformularios](#) (sólo subformularios lista) - [Áreas 4D Write Pro](#)

# Rango de valores

## Valor por defecto

Puede asignar un valor por defecto para ser introducido en un objeto de entrada. Esta propiedad es útil, por ejemplo, cuando la entrada [fuente de datos](#) es un campo: el valor por defecto se introduce cuando se muestra un nuevo registro por primera vez. Puede cambiar el valor a menos que el área de entrada se haya definido como [no editable](#).

El valor por defecto sólo puede utilizarse si el [tipo de fuente de datos](#) es:

- texto/cadena
- number/integer
- fecha
- time
- booleano

4D ofrece sellos para generar valores por defecto para la fecha, la hora y el número de secuencia. La fecha y la hora se toman de la fecha y la hora del sistema. 4D genera automáticamente los números de secuencia necesarios. La siguiente tabla muestra el sello a utilizar para generar valores por defecto de forma automática:

Sello	Significado
#D	Fecha actual
#H	Hora actual
#N	Número de secuencia

Puede utilizar un número de secuencia para crear un número único para cada registro de la tabla para el archivo de datos actual. Un número de secuencia es un longint que se genera para cada nuevo registro. Los números comienzan en uno (1) y van aumentando de uno en uno. Un número de secuencia no se repite nunca, incluso si el registro al que se asigna se elimina de la tabla. Cada tabla tiene su propio contador interno de números de secuencia. Para más información, consulte el párrafo [Autoincremento](#).

No hay que confundir esta propiedad con la propiedad "valores por defecto" que permite llenar una columna list box con valores estáticos.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
defaultValue	string, number, date, time, boolean	Todo valor y/o un sello: "#D", "#H", "#N"

## Objetos soportados

### Entrada

## Lista de excluidos

Permite definir una lista cuyos valores no pueden introducirse en el objeto. Si se introduce un valor excluido, no se acepta y se muestra un mensaje de error.

Si una lista especificada es jerárquica, sólo se tienen en cuenta los elementos del primer nivel.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
excludedList	lista	Una lista de valores a excluir.

Objetos soportados

Combo Box - Columna List Box - Entrada

---

## Lista requerida

Restringe las entradas válidas a los elementos de la lista. Por ejemplo, es posible que desee utilizar una lista obligatoria para los títulos de los puestos de trabajo, de modo que las entradas válidas se limiten a los títulos que han sido aprobados por la dirección.

La creación de una lista obligatoria no muestra automáticamente la lista cuando se selecciona el campo. Si desea mostrar la lista requerida, asigne la misma lista a la propiedad [Lista de opciones](#). Sin embargo, a diferencia de la propiedad [Lista de selección](#), cuando se define una lista requerida, ya no es posible la introducción mediante el teclado, sólo se permite la selección de un valor de la lista mediante el menú emergente. Si se definen diferentes listas utilizando las propiedades [Lista de selección](#) y Lista requerida, la propiedad Lista requerida tiene prioridad.

Si una lista especificada es jerárquica, sólo se tienen en cuenta los elementos del primer nivel.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
requiredList	lista	Una lista de valores obligatorios.

Objetos soportados

Combo Box - Columna List Box - Entrada

# Opciones de redimensionamiento

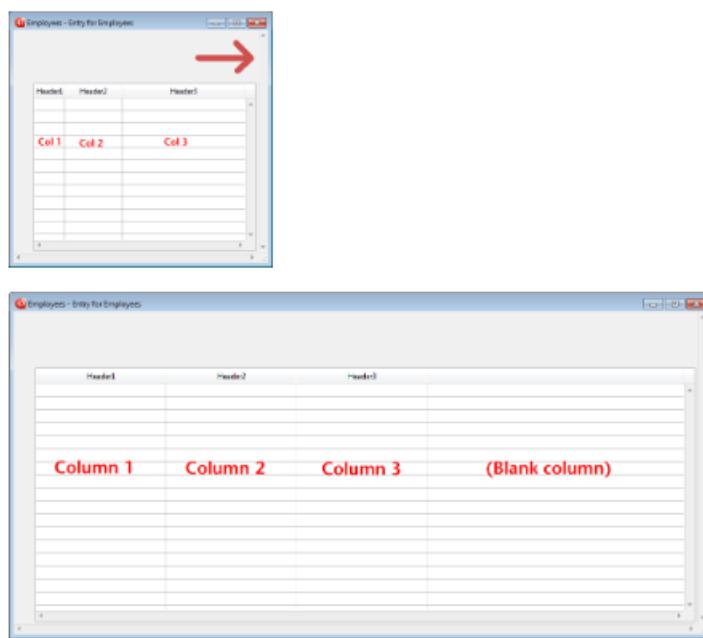
## Redimensionamiento columnas auto

Cuando esta propiedad está activa (valor `rightToLeft` en JSON), las columnas del listbox se redimensionan automáticamente junto con el list box, dentro de los límites de los anchos **mínimo** y **máximo** definidos.

Cuando esta propiedad está desactivada (valor `legacy` en JSON), sólo se redimensiona la columna más a la derecha del listbox, aunque su ancho supere el valor máximo definido.

### Cómo funciona el redimensionamiento automático de las columnas

- A medida que el ancho del list box aumenta, sus columnas se amplían, una a una, empezando de derecha a izquierda, hasta que cada una alcanza su **ancho máximo**. Sólo se redimensionan las columnas con la propiedad **Resizable** seleccionada.
- El mismo procedimiento se aplica cuando el ancho del list box disminuye, pero en orden inverso (es decir, las columnas se redimensionan empezando de izquierda a derecha). Cuando cada columna ha alcanzado su **ancho mínimo**, la barra de desplazamiento horizontal vuelve a activarse.
- Las columnas se redimensionan sólo cuando la barra de desplazamiento horizontal no está "activa"; es decir, todas las columnas son totalmente visibles en el list box en su tamaño actual. Nota: si la barra de desplazamiento horizontal está oculta, esto no altera su estado: una barra de desplazamiento puede seguir estando activa, aunque no sea visible.
- Una vez que todas las columnas alcanzan su tamaño máximo, dejan de ampliarse y en su lugar se añade una columna en blanco (falsa) a la derecha para llenar el espacio extra. Si hay una columna falsa (en blanco), cuando el ancho del list box disminuye, ésta es la primera área que se reduce.



### Sobre la columna falsa (en blanco)

La apariencia de la columna falsa coincide con la de las columnas existentes; tendrá un encabezado y/o un pie de página falsos si estos elementos están presentes en las columnas del list box existentes y tendrá aplicados los mismos colores de fondo.

Se puede hacer clic en el encabezado y/o en el pie de página falsos, pero esto no tiene ningún efecto sobre las otras columnas (por ejemplo: no se realiza ninguna ordenación); no obstante, los eventos se generan en consecuencia `On`

`Clicked`, `On Header Click` y `On Footer Click` se generan en consecuencia.

Si se hace clic en una celda de la columna falsa, el comando [LISTBOX GET CELL POSITION](#) devuelve "X+1" para su número de columna (donde X es el número de columnas existentes).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
resizingMode	cadena	"rightToLeft", "legacy"

## Objetos soportados

### List Box

## Dimensionamiento horizontal

Esta propiedad indica si el tamaño horizontal de un objeto debe ser movido o redimensionado cuando un usuario cambia el tamaño del formulario. También puede definirse dinámicamente por el comando del lenguaje [OBJECT SET RESIZING OPTIONS](#).

Hay tres opciones disponibles:

Option	Valor JSON	Resultado
Agrandar	"grow"	El mismo porcentaje se aplica al ancho del objeto cuando el usuario redimensiona el ancho de la ventana,
Mover	"move"	El objeto se desplaza la misma cantidad a la izquierda o a la derecha que el aumento del ancho cuando el usuario redimensiona el ancho de la ventana,
Ninguno	"fixed"	El objeto permanece inmóvil cuando se cambia el tamaño del formulario

Esta propiedad funciona junto con la propiedad [Dimensionamiento vertical](#).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
sizingX	cadena	"grow", "move", "fixed"

## Objetos soportados

[Área 4D View Pro](#) - [Área 4D Write Pro](#) - [Botón](#) - [Rejilla de botones](#) - [Casilla de selección](#) - [Combo Box](#) - [Lista desplegable](#) - [Group Box](#) - [Lista jerárquica](#) - [Área de entrada](#) - [List Box](#) - [Línea](#) - [Columna List Box](#) - [Óvalo](#) - [Botón imagen](#) - [Menú emergente de imagen](#) - [Área de plugins](#) - [Indicadores de progreso](#) - [Botón radio](#) - [Regla](#) - [Rectángulo](#) - [Spinner](#) - [Splitter](#) - [Imagen estática Stepper](#) - [Sub-formulario](#) - [Pestaña](#) - [Área Web](#)

## Dimensionamiento vertical

Esta propiedad indica si el tamaño vertical de un objeto debe ser movido o redimensionado cuando un usuario cambia el tamaño del formulario. También puede definirse dinámicamente por el comando del lenguaje [OBJECT SET RESIZING OPTIONS](#).

Hay tres opciones disponibles:

Option	Valor JSON	Resultado
Agrandar	"grow"	El mismo porcentaje se aplica a la altura del objeto cuando el usuario redimensiona el ancho de la ventana,
Mover	"move"	El objeto se desplaza la misma cantidad hacia arriba o hacia abajo que el aumento de la altura cuando el usuario redimensiona el ancho de la ventana,
Ninguno	"fixed"	El objeto permanece inmóvil cuando se cambia el tamaño del formulario

Esta propiedad funciona junto con la propiedad [Dimensionamiento horizontal](#).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
sizingY	cadena	"grow", "move", "fixed"

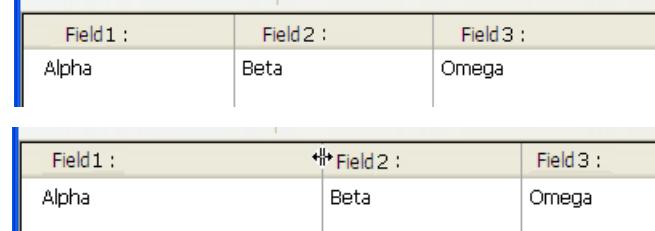
## Objetos soportados

[Área 4D View Pro](#) - [Área 4D Write Pro](#) - [Botón](#) - [Rejilla de botones](#) - [Casilla de selección](#) - [Combo Box](#) - [Lista desplegable](#) - [Group Box](#) - [Lista jerárquica](#) - [Área de entrada](#) - [List Box](#) - [Línea](#) - [Columna List Box](#) - [Óvalo](#) - [Botón imagen](#) - [Menú emergente de imagen](#) - [Área de plugins](#) - [Indicadores de progreso](#) - [Botón radio](#) - [Regla](#) - [Rectángulo](#) - [Spinner](#) - [Splitter](#) - [Imagen estática Stepper](#) - [Sub-formulario](#) - [Pestaña](#) - [Área Web](#)

## Pulsador

Cuando un objeto splitter tiene esta propiedad, los otros objetos a su derecha (splitter vertical) o debajo de él (separador horizontal) son empujados al mismo tiempo que el splitter, sin parar.

Este es el resultado del movimiento de un splitter “pusher”:



Cuando esta propiedad no se aplica al splitter, el resultado es el siguiente:



## Gramática JSON

Nombre	Tipos de datos	Valores posibles
splitterMode	cadena	"move" (pusher), "resize" (standard)

## Objetos soportados

[Separador](#)

## Redimensionable

Designa si el tamaño de la columna puede ser modificado por el usuario.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
redimensionable	booleano	"true", "false"

Objetos soportados

[Columna de list box](#)

# Escala

---

## Barber shop

Activa la variante "barber shop" para el termómetro.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
max	number	NO pasado = activado; pasado = desactivado (termómetro básico)

Objetos soportados

[Barber shop](#)

---

## Mostrar graduación

Muestra/Oculta las graduaciones junto a las etiquetas.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
showGraduations	booleano	"true", "false"

Objetos soportados

[Termómetro - Regla](#)

---

## Paso en la graduación

Medición de la visualización de la escala.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
graduationStep	integer	mínimo: 0

Objetos soportados

[Termómetro - Regla](#)

---

## Posición de la etiqueta

Especifica la ubicación del texto mostrado de un objeto.

- Ninguno - no se muestra ninguna etiqueta
- Arriba - Muestra las etiquetas a la izquierda o sobre el indicador

- Abajo - Muestra las etiquetas a la derecha o debajo de un indicador

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
labelsPlacement	cadena	"none", "top", "bottom", "left", "right"

Objetos soportados

[Termómetro - Regla](#)

---

## Máximo

Valor máximo de un indicador.

- Para los steppers numéricos, esta propiedad representa los segundos cuando el objeto está asociado a un valor de tipo hora y se ignoran cuando están asociados a un valor de tipo fecha.
- Para activar los [termómetros del Barber Shop](#), esta propiedad debe omitirse.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
max	string / number	mínimo: 0 (para los tipos de datos numéricos)

Objetos soportados

[Termómetro - Regla - Stepper](#)

---

## Mínimo

Valor mínimo de un indicador. Para los steppers numéricos, esta propiedad representa los segundos cuando el objeto está asociado a un valor de tipo hora y se ignoran cuando están asociados a un valor de tipo fecha.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
min	string / number	mínimo: 0 (para los tipos de datos numéricos)

Objetos soportados

[Termómetro - Regla - Stepper](#)

---

## Step

Intervalo mínimo aceptado entre los valores durante el uso. Para los steppers numéricos, esta propiedad representa los segundos cuando el objeto está asociado a un valor de tipo hora y los días cuando está asociado a un valor de tipo fecha.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
step	integer	mínimo: 1

Objetos soportados

[Termómetro](#) - [Regla](#) - [Stepper](#)

# Subformulario

---

## Autorizar la eliminación

Especifica si el usuario puede eliminar subregistros en un subformulario listado.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
deletableInList	booleano	true, false (por defecto: true)

Objetos soportados

[Subformulario](#)

---

## Formulario detallado

Esta propiedad se utiliza para declarar el formulario detallado que se utilizará en el subformulario. Puede ser:

- un widget, es decir, un subformulario de tipo página dotado de funciones específicas. En este caso, las propiedades [subformulario de lista](#) y [Fuente](#) deben estar vacías o no estar presentes.  
Se puede seleccionar el nombre de un formulario de componente cuando se publica en el componente.

Puede generar [componentes](#) que den funcionalidades adicionales a través de subformularios.

- el formulario detallado a asociar al [subformulario listado](#). El formulario detallado puede utilizarse para introducir o ver los subregistros. Generalmente contiene más información que el formulario lista. Naturalmente, el formulario detallado debe pertenecer a la misma tabla que el subformulario. Normalmente se utiliza un formulario de salida como formulario lista y un formulario de entrada como formulario detallado. Si no especifica el formulario a utilizar para la entrada de la página completa, 4D utiliza automáticamente el formato de entrada por defecto de la tabla.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
detailForm	cadena	Nombre (cadena) de la tabla o formulario proyecto, una ruta POSIX (cadena) a un archivo .json que describa el formulario, o un objeto que describa el formulario

Objetos soportados

[Subformulario](#)

---

## Doble clic en línea vacía

Acción a realizar en caso de doble clic en una línea vacía de un subformulario listado. Las siguientes opciones están disponibles:

- No hacer nada: ignora el doble clic.
- Añadir registro: crea un nuevo registro en el subformulario y cambia al modo edición. El registro se creará directamente en la lista si la propiedad [Editable en la lista] está activada. En caso contrario, se creará en modo página, en el [formulario detallado](#) asociado al subformulario.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
doubleClickInEmptyAreaAction	cadena	"addSubrecord" o "" to do nothing

Objetos soportados

[Subformulario](#)

Ver también

[Doble clic en línea](#)

---

## Doble clic en línea

### Sub-formularios lista

Define la acción a realizar cuando un usuario haga doble clic en una línea en un subformulario lista. Las opciones disponibles son:

- No hacer nada (por defecto): hacer doble clic en una línea no desencadena ninguna acción automática.
- Editar registro: al hacer doble clic en una línea se muestra el registro correspondiente en el [formulario detallado](#) definido para el subformulario lista. El registro se abre en modo de lectura-escritura para que pueda ser modificado.
- Mostrar registro: idéntica a la acción anterior, salvo que el registro se abre en modo de sólo lectura para que no pueda ser modificado.

Independientemente de la acción seleccionada/elegida, se genera el evento de formulario `On Double clicked`.

Para las dos últimas acciones, también se genera el evento de formulario `On Open Detail`. `On Close Detail` se genera cuando un registro mostrado en el formulario detallado asociado al list box está a punto de cerrarse (independientemente de que el registro se haya modificado o no).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
doubleClickInRowAction	cadena	"editSubrecord", "displaySubrecord"

Objetos soportados

[Subformulario](#)

Ver también

[Doble clic en línea vacía](#)

---

## Editable en lista

Cuando un subformulario lista tiene esta propiedad activada, el usuario puede modificar los datos del registro directamente en la lista, sin tener que utilizar el [formulario detallado asociado](#).

Para ello, basta con hacer dos clics en el campo a modificar para que pase al modo edición (asegúrese de dejar

suficiente tiempo entre los dos clics para no generar un doble clic).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
enterableInList	booleano	true, false

Objetos soportados

[Subformulario](#)

---

## Formulario listado

Esta propiedad se utiliza para declarar el formulario listado que se utilizará en el subformulario. Un subformulario lista le permite introducir, ver y modificar datos en otras tablas.

Los subformularios de lista pueden utilizarse para la entrada de datos de dos maneras: el usuario puede introducir los datos directamente en el subformulario, o introducirlos en un [formulario de entrada](#). En esta configuración, el formulario utilizado como subformulario se denomina formulario Lista. El formulario de entrada se denomina formulario detallado.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
listForm	cadena	Nombre (cadena) de la tabla o formulario proyecto, una ruta POSIX (cadena) a un archivo .json que describa el formulario, o un objeto que describa el formulario

Objetos soportados

[Subformulario](#)

---

## Source

Especifica la tabla a la que pertenece el subformulario Lista (si la hay).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
tabla	cadena	Nombre de la tabla 4D, o "" si no hay tabla.

Objetos soportados

[Subformulario](#)

---

## Modo de selección

Designa la opción para permitir a los usuarios seleccionar líneas:

- Ninguna: las líneas no se pueden seleccionar si se elige este modo. Hacer clic en la lista no tendrá ningún efecto a menos que la opción [Editable en lista](#) esté activada. Las teclas de navegación sólo hacen que la lista se desplace; no se genera el evento de formulario `On Selection Change`.

- Simple: en este modo se puede seleccionar una línea a la vez. Si hace clic en una línea, la seleccionará. Un Ctrl+clic (Windows) o Comando+clic (macOS) en una línea cambia su estado (entre seleccionada o no). Las teclas de flecha arriba y abajo seleccionan la línea anterior/siguiente de la lista. Las otras teclas de navegación se desplazan por la lista. El evento de formulario `On Selection Change` se genera cada vez que se cambia la línea actual.
- Múltiple: en este modo se pueden seleccionar varias líneas simultáneamente.
  - Los subregistros seleccionados son devueltos por el comando `GET HIGHLIGHTED RECORDS`.
  - Al hacer clic en el registro se selecciona, pero no se modifica el registro actual.
  - Un Ctrl+clic (Windows) o Comando+clic (macOS) en un registro cambia su estado (entre seleccionado o no). Las teclas de flecha arriba y abajo seleccionan el registro anterior/siguiente en la lista. Las otras teclas de navegación se desplazan por la lista. El evento de formulario `On Selection Change` se genera cada vez que el registro seleccionado se modifica.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
selectionMode	cadena	"multiple", "single", "none"

## Objetos soportados

[Subformulario](#)

# Texto

---

## Autorizar selector fuente/color

Cuando esta propiedad está activa, los comandos [OPEN FONT PICKER](#) y [OPEN COLOR PICKER](#) pueden ser llamados para mostrar las ventanas de selección de fuentes y colores del sistema. A través de estas ventanas, los usuarios pueden cambiar la fuente o el color de un objeto formulario que tenga el foco directamente haciendo clic. Cuando esta propiedad está desactivada (por defecto), los comandos del selector abierto no tienen efecto.

Gramática JSON

Propiedad	Tipos de datos	Valores posibles
allowFontColorPicker	booleano	false (por defecto), true

Objetos soportados

[Entrada](#)

---

## Negrita

Ajusta el texto seleccionado para que aparezca más oscuro y pesado.

También puede definir esta propiedad utilizando el comando [OBJECT SET FONT STYLE](#).

Este es un texto normal.  
Este es un texto en negrita.

Gramática JSON

Propiedad	Tipos de datos	Valores posibles
fontWeight	texto	"normal", "bold"

Objetos soportados

[Botón](#) - [Casilla de selección](#) - [Combo Box](#) - [Lista desplegable](#) - [Group Box](#) - [Lista jerárquica](#) - [Área de entrada](#) - [List Box](#) - [Columna List Box](#) - [Pie List Box](#) - [Encabezado List Box](#) - [Botón radio](#) - [Área de texto](#)

---

## Itálica

Hace que el texto seleccionado se incline ligeramente hacia la derecha.

También puede definir esta propiedad utilizando el comando [OBJECT SET FONT STYLE](#).

Este es un texto normal.  
*Este es un texto en itálica.*

Gramática JSON

Nombre	Tipos de datos	Valores posibles
fontStyle	cadena	"normal", "italic"

## Objetos soportados

Botón - Casilla de selección - Combo Box - Lista desplegable - Group Box - Lista jerárquica - Área de entrada - List Box - Columna List Box - Pie List Box - Encabezado List Box - Botón radio - Área de texto

---

## Subrayado

Hace que el texto tenga una línea por debajo.

Este es un texto normal.  
Este es un texto subrayado.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
textDecoration	cadena	"normal", "underline"

## Objetos soportados

Botón - Casilla de selección - Combo Box - Lista desplegable - Group Box - Lista jerárquica - Área de entrada - List Box - Columna List Box - Pie List Box - Encabezado List Box - Botón radio - Área de texto

---

## Fuente

Esta propiedad permite indicar el tema de la fuente o la familia de fuente utilizada en el objeto.

Las propiedades Tema de la fuente y de la familia de la fuente son mutuamente excluyentes. Un tema de fuente se encarga de los atributos de fuente, incluido el tamaño. Una familia de fuentes permite definir el nombre, el tamaño y el color de la fuente.

### Tema de fuente

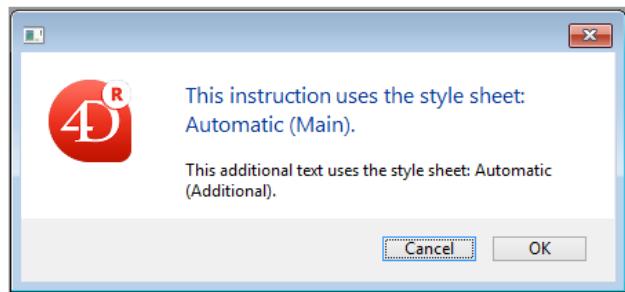
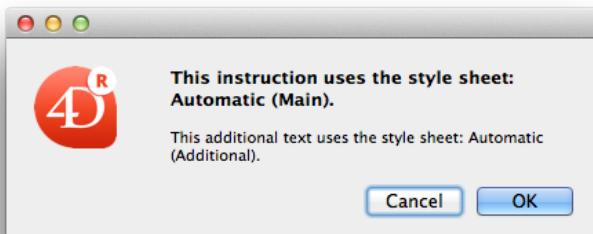
La propiedad de tema de fuente designa un nombre de estilo automático. Los estilos automáticos determinan de forma dinámica la familia de fuentes, el tamaño y el color de la fuente que se utilizará para el objeto, según los parámetros sistema. Estos parámetros dependen de:

- la plataforma,
- el lenguaje del sistema,
- y el tipo de objeto de formulario.

Con el tema de fuente, se garantiza que los títulos se muestren siempre de acuerdo con los estándares actuales de la interfaz del sistema. Sin embargo, su tamaño puede variar de una máquina a otra.

Hay tres temas de fuentes disponibles:

- normal: estilo automático, aplicado por defecto a todo nuevo objeto creado en el editor de formularios.
- Los temas de fuentes principales y suplementarios solo son soportados por las [áreas de texto](#) y las [áreas de entrada](#). Estos temas están pensados principalmente para diseñar cajas de diálogo. Se refieren a los estilos de fuente utilizados, respectivamente, para el texto principal y la información adicional en las ventanas de su interfaz. A continuación se muestran las cajas de diálogo típicas (macOS y Windows) que utilizan estos temas de fuentes:



Los temas de fuentes gestionan la fuente, así como su tamaño y color. You can apply custom style properties (Bold, Italic or Underline) without altering its functioning.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
fontTheme	cadena	"normal", "main", "additional"

## Objetos soportados

[Botón](#) - [Casilla de selección](#) - [Combo Box](#) - [Lista desplegable](#) - [Group Box](#) - [Lista jerárquica](#) - [Área de entrada](#) - [List Box](#) - [Columna List Box](#) - [Pie List Box](#) - [Encabezado List Box](#) - [Botón radio](#) - [Área de texto](#)

## Familia de fuentes

Hay dos tipos de nombres de familias de fuentes:

- *family-name*: El nombre de una familia de fuentes, como "times", "courier", "arial", etc.
- *generic-family*: El nombre de una familia genérica, como "serif", "sans-serif", "cursive", "fantasy", "monospace".

Puede definirla utilizando el comando [OBJECT SET FONT](#).

This is Times New Roman font.

This is Calibri font.

This is Papyrus font.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
fontFamily	cadena	Nombre de la familia de fuentes CSS

4D recomienda utilizar sólo fuentes [seguras para la web](#).

## Objetos soportados

[Botón](#) - [Casilla de selección](#) - [Combo Box](#) - [Lista desplegable](#) - [Group Box](#) - [Lista jerárquica](#) - [Área de entrada](#) - [List Box](#) - [Columna List Box](#) - [Pie List Box](#) - [Encabezado List Box](#) - [Botón radio](#) - [Área de texto](#)

## Tamaño fuente

Permite definir el tamaño de la fuente del objeto en puntos.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
fontSize	integer	Tamaño de letra en puntos. Valor mínimo: 0

### Objetos soportados

Botón - Casilla de selección - Combo Box - Lista desplegable - Group Box - Lista jerárquica - Área de entrada - List Box - Columna List Box - Pie List Box - Encabezado List Box - Botón radio - Área de texto

---

## Color de fuente

Designa el color de la fuente.

Esta propiedad también define el color de [borde](#) (si lo hay) del objeto cuando se utiliza el estilo "plano" o "punteado".

El color puede ser especificado por:

- un nombre de color - como "red"
- un valor HEX - como "# ff0000"
- un valor RVB - como "rgb (255,0,0)"

También puede definir esta propiedad utilizando el comando [OBJECT SET RGB COLORS](#).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
stroke	cadena	un valor css, "transparent", "automatic"

### Objetos soportados

Botón - Casilla de selección - Combo Box - Lista desplegable - Group Box - Lista jerárquica - Área de entrada - List Box - Columna List Box - Pie List Box - Encabezado List Box - Indicadores de progreso - Regla - Botón Radio - Área de texto

---

## Expresión color fuente

List box de tipo colección/selección de entidades

Se utiliza para aplicar un color de fuente personalizado a cada línea del list box. Debe utilizar valores de color RGB. Para más información al respecto, consulte la descripción del comando [OBJECT SET RGB COLORS](#) en el manual Lenguaje de 4D.

Debe introducir una expresión o una variable (no se pueden utilizar variables de tipo array). La expresión o variable se evaluará para cada línea mostrada. Puede utilizar las constantes del tema [SET RGB COLORS](#).

También puede definir esta propiedad utilizando el comando [LISTBOX SET PROPERTY](#) con la constante `lk font color expression`.

Esta propiedad también puede definirse mediante una [Expresión Meta Info](#).

El siguiente ejemplo utiliza un nombre de variable: introduzca `CompanyColor` para la Expresión color fuente y, en el método formulario, escriba el siguiente código:

```
CompanyColor:=Choose([Companies]ID;Background color;Light shadow color;  
Foreground color;Dark shadow color)
```

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
rowStrokeSource	cadena	Expresión color fuente

## Objetos soportados

[List Box](#)

---

## Expresión estilo

```
List box de tipo colección/selección de entidades
```

Utilizado para aplicar un estilo de fuente personalizado a cada línea de list box o de cada celda de la columna.

Debe introducir una expresión o una variable (no se pueden utilizar variables de tipo array). La expresión o variable se evaluará para cada línea mostrada (si se aplica al list box) o cada celda mostrada (si se aplica a una columna). Puede utilizar las constantes del tema [Estilos de fuentes](#).

Ejemplo:

```
Choose([Companies]ID;Bold;Plain;Italic;Underline)
```

También puede definir esta propiedad utilizando el comando `LISTBOX SET PROPERTY` con la constante `lk font style expression`.

Esta propiedad también puede definirse mediante una [Expresión Meta Info](#).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
rowStyleSource	cadena	Expresión de estilo a evaluar para cada línea/celda.

## Objetos soportados

[List Box - Columna List Box](#)

---

## Alineación horizontal

Ubicación horizontal del texto dentro del área que lo contiene.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
textAlign	cadena	"automatic", "right", "center", "justify", "left"

## Objetos soportados

## Alineamiento vertical

Ubicación vertical del texto dentro del área que lo contiene.

La opción Predeterminado ( `automático` valor JSON) define la alineación según el tipo de datos que se encuentran en cada columna:

- `abajo` para todos los datos (excepto las imágenes) y
- `arriba` para los datos del tipo imagen.

Esta propiedad también puede ser manejada por los comandos [OBJECT Get vertical alignment](#) y [OBJECT SET VERTICAL ALIGNMENT](#).

Gramática JSON

Nombre	Tipos de datos	Valores posibles
verticalAlign	cadena	"automatic", "top", "middle", "bottom"

Objetos soportados

[List Box](#) - [Columna List Box](#) - [Pie de List Box](#) - [Encabezado List Box](#)

---

## Meta Info expression

`List box de tipo colección o entity selection (selección de entidades)`

Indica una expresión o una variable que se evaluará para cada línea mostrada. Permite definir todo un conjunto de atributos texto de las líneas. Debe pasar una variable objeto o una expresión que devuelva un objeto . Se soportan las siguientes propiedades:

Nombre de propiedad	Tipo	Descripción
stroke	cadena	Color de la fuente. Todo color CSS (por ejemplo: "#FF00FF"), "automatic", "transparent"
fill	cadena	Color de fondo. Todo color CSS (por ejemplo: "#F00FFF"), "automatic", "transparent"
fontStyle	cadena	"normal","italic"
fontWeight	cadena	"normal","bold"
textDecoration	cadena	"normal","underline"
unselectable	booleano	Designa la línea correspondiente como no seleccionable (* es decir, *, no es posible el resaltado). Las áreas que se pueden introducir ya no se pueden introducir si esta opción está activada, a menos que la opción "Edición con un solo clic" también esté activada. Los controles como las casillas de selección y las listas siguen siendo funcionales. This setting is ignored if the list box selection mode is "None". Valores por defecto: False.
disabled	booleano	Desactiva la línea correspondiente. Las áreas editables ya no son accesibles si esta opción está activada. El texto y los controles (casillas de selección, listas, etc.) aparecen atenuados o en gris. Valores por defecto: False.
cell. <columnName>	objeto	Permite aplicar la propiedad a una sola columna. Pase en <columnName> el nombre del objeto de la columna de list box. Nota: las propiedades "no seleccionable" y "desactivada" sólo pueden definirse a nivel de la línea. Se ignoran si se pasan en el objeto "celda"

Los parámetros de estilo definidos con esta propiedad se ignoran si ya se han definido otros parámetros de estilo mediante expresiones (\*es decir, \*, [Style Expression](#), [Font Color Expression](#), [Background Color Expression](#)).

## Ejemplo

En el método proyecto *Color*, escribe el siguiente código:

```
//Método Color
//Define el color de la fuente para ciertas líneas y el color de fondo para una columna específica: C_0B
Form.meta:=New object
If(This.ID>5) //ID es un atributo de objetos/entidades de una colección
  Form.meta.stroke:="purple"
  Form.meta.cell:=New object("Column2";New object("fill";"black"))
Else
  Form.meta.stroke:="orange"
End if
$0:=Form.meta
```

Buenas prácticas: por razones de optimización, se recomendaría en este caso crear el objeto `meta.cell` una vez en el método formulario:

```
//método formulario
Case of
  :(Form event code=On Load)
    Form.colStyle:=New object("Column2";New object("fill";"black"))
End case
```

Entonces, el método *Color* contendría:

```

//método Color
...
If(This.ID>5)
  Form.meta.stroke:="purple"
  Form.meta.cell:=Form.colStyle //reutilizar el mismo objeto para mejorar el rendimiento
...

```

Ver también el comando [This](#).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
metaSource	cadena	Expresión de objeto a evaluar para cada línea/celda.

## Objetos soportados

[List Box](#)

---

## Multistyle

Esta propiedad permite la posibilidad de utilizar estilos específicos en el área seleccionada. When this option is checked, 4D interprets any `<SPAN>` HTML tags found in the area.

Por defecto, esta opción no está activa.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
styledText	booleano	true, false

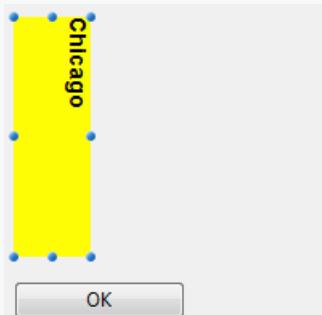
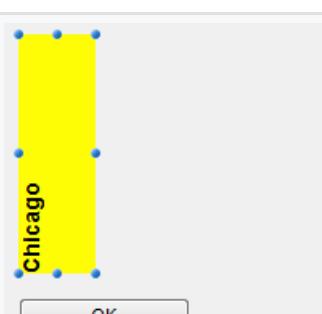
## Objetos soportados

[List Box Column - Input](#)

---

## Orientación

Modifica la orientación (rotación) de un área de texto. Las áreas de texto pueden girarse en incrementos de 90°. Cada valor de orientación se aplica manteniendo el mismo punto de partida inferior izquierdo para el objeto:

Valor de orientación	Resultado
0 (por defecto)	
90	
180	
270	

Además de [áreas de texto estáticas](#), los objetos de texto de las [áreas de entrada](#) pueden girar cuando no son [editables](#). Cuando se aplica una propiedad de rotación a un objeto de entrada, se elimina la propiedad editable (si la hay). Este objeto se excluye entonces del orden de entrada.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
textAngle	number	0, 90, 180, 270

#### Objetos soportados

[Entrada](#) (no editable) - [Área de texto](#)

## Array colores de fuente

#### List box de tipo array

Permite definir un color de fuente personalizado para cada línea del list box o celda de la columna.

Se debe utilizar el nombre de un array Entero largo. Cada elemento de este array corresponde a una línea del list box (si se aplica al list box) o a una celda de la columna (si se aplica a una columna), por lo que el array debe tener el mismo tamaño que el array asociado a la columna. Puede utilizar las constantes del tema [SET RGB COLORS](#). Si desea que la celda herede el color de fondo definido en el nivel superior, pase el valor -255 al elemento del array correspondiente.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
rowStrokeSource	cadena	El nombre de un array entero largo

#### Objetos soportados

[List Box - Columna List Box](#)

---

## Array de estilos

#### List box de tipo array

Permite definir un estilo de fuente personalizado para cada línea del list box o cada celda de la columna.

Se debe utilizar el nombre de un array Entero largo. Cada elemento de este array corresponde a una línea del list box (si se aplica al list box) o a una celda de la columna (si se aplica a una columna), por lo que el array debe tener el mismo tamaño que el array asociado a la columna. Para llenar el array (utilizando un método), utilice las constantes del tema [Estilos de fuente](#). Se pueden añadir constantes para combinar estilos. Si desea que la celda herede el estilo definido en el nivel superior, pase el valor -255 al elemento del array correspondiente.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
rowStyleSource	cadena	El nombre de un array entero largo.

#### Objetos soportados

[List Box - Columna List Box](#)

---

## Almacenar con etiquetas de estilo por defecto

Esta propiedad sólo está disponible para un área de entrada [Multi-estilo](#). Cuando esta propiedad está activada, el área almacenará las etiquetas de estilo con el texto, incluso si no se ha realizado ninguna modificación. En este caso, las etiquetas corresponden al estilo por defecto. Cuando esta propiedad está desactivada, sólo se almacenan las etiquetas de estilo modificadas.

Por ejemplo, este es un texto que incluye una modificación de estilo:

What a beautiful day

Cuando la propiedad está desactivada, el área sólo almacena la modificación. Por lo tanto, los contenidos almacenados son:

i Qué <SPAN style="font-size:13.5pt">hermoso</SPAN> día!

Cuando la propiedad está activa, el área almacena toda la información de formato. La primera etiqueta genérica describe el estilo por defecto y luego cada variación es objeto de un par de etiquetas anidadadas. Por lo tanto, los contenidos almacenados en el área son:

```
<SPAN style="font-family:'Arial';font-size:9pt;text-align:left;font-weight:normal;font-style:normal;text
```

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
storeDefaultStyle	booleano	true, false (por defecto).

## Objetos soportados

[Entrada](#)

# Texto e Imagen

---

## Ruta de acceso fondo

Define la ruta de la imagen que se dibujará en el fondo del objeto. Si el objeto utiliza un icono con diferentes estados, la imagen de fondo soportará automáticamente el mismo número de estados.

El nombre de la ruta a introducir es similar al de la propiedad [Ruta de acceso para las imágenes estáticas](#).

Gramática JSON

Nombre	Tipos de datos	Valores posibles
customBackgroundPicture	cadena	Ruta relativa en sintaxis POSIX. Debe utilizarse junto con la opción "Personalizado" de la propiedad "Style".

Objetos soportados

[Botón personalizado](#) - [Casilla de selección personalizada](#) - [Botón radio personalizado](#)

---

## Estilos de botón

Aspecto general del botón. El estilo del botón también influye en la disponibilidad de ciertas opciones.

Gramática JSON

Nombre	Tipos de datos	Valores posibles
style	texto	"regular", "flat", "toolbar", "bevel", "roundedBevel", "gradientBevel", "texturedBevel", "office", "help", "circular", "disclosure", "roundedDisclosure", "custom"

Objetos soportados

[Botón](#) - [Botón radio](#) - [Casilla de selección](#) - [Botón Radio](#)

---

## Margen horizontal

Esta propiedad permite definir el tamaño (en píxeles) de los márgenes horizontales del botón. Este margen delimita el área que el ícono del botón y el título no deben sobrepasar.

Este parámetro es útil, por ejemplo, cuando la imagen de fondo contiene bordes:

Con / Sin	Ejemplo
Sin margen	
Con un margen 13 píxeles	

Esta propiedad funciona junto con la propiedad [Margen vertical](#).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
customBorderX	number	Para usar con el estilo "personalizado". Mínimo: 0

## Objetos soportados

[Botón personalizado](#) - [Casilla de selección personalizada](#) - [Botón radio personalizado](#)

---

## Ubicación del ícono

Designa la ubicación de un ícono en relación con el objeto formulario.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
iconPlacement	cadena	"ninguno", "izquierda", "derecha"

## Objetos soportados

[Encabezado de list box](#)

---

## Desplazamiento ícono

Define un valor de desplazamiento personalizado en píxeles, que se utilizará cuando se haga clic en el botón

El título del botón se desplazará hacia la derecha y hacia la parte inferior por el número de píxeles introducidos. Esto permite aplicar un efecto 3D personalizado cuando se presiona el botón.

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
customOffset	number	mínimo: 0

## Objetos soportados

[Botón personalizado](#) - [Casilla de selección personalizada](#) - [Botón radio personalizado](#)

---

## Número de estados

Esta propiedad define el número exacto de estados presentes en la imagen utilizada como ícono para un [botón con ícono](#), una [casilla de selección](#) o un [botón radio](#) personalizado. En general, el ícono de un botón incluye cuatro estados: activo, presionado, sobre el ratón e inactivo.

Cada estado está representado por una imagen diferente. En la imagen fuente, los estados deben apilarse verticalmente:



Están representados los siguientes estados:

1. botón no presionado / casilla de selección no marcada (valor de la variable=0)
2. botón presionado / casilla de selección marcada (valor variable=1)
3. pasar por encima
4. disabled

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
iconFrames	number	mínimo: 1

#### Objetos soportados

[Botón](#) (todos los estilos excepto [Ayuda](#)) - [Casilla de selección](#) - [Botón radio](#)

---

## Ruta de acceso de la imagen

Define la ruta de la imagen que se utilizará como ícono del objeto.

El nombre de la ruta a introducir es similar al de [la propiedad Ruta de acceso para las imágenes estáticas](#).

Cuando se utiliza como ícono de objetos activos, la imagen debe estar diseñada para soportar un [número de estados](#) variable.

#### Gramática JSON

Nombre	Tipos de datos	Valores posibles
icon	imagen	Ruta relativa o filesystem en sintaxis POSIX.

#### Objetos soportados

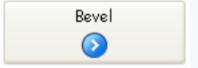
[Botón](#) (todos los estilos excepto [Ayuda](#)) - [Casilla de selección](#) - [Encabezado List Box](#) - [Botón radio](#)

---

## Posición título/imagen

Esta propiedad permite modificar la ubicación relativa del título del botón en relación con el ícono asociado. Esta propiedad no tiene efecto cuando el botón sólo contiene un título (sin ícono asociado) o una ícono (sin título). Por defecto, cuando un botón contiene un título y una ícono, el texto se coloca debajo de la ícono.

Aquí están los resultados utilizando las distintas opciones para esta propiedad:

Option	Descripción	Ejemplo
Izquierda	El texto se coloca a la izquierda del ícono. El contenido del botón se alinea a la derecha.	
Arriba	El texto se coloca sobre el ícono. El contenido del botón está centrado.	
Derecha	El texto se coloca a la derecha del ícono. El contenido del botón se alinea a la izquierda.	
Abajo	El texto se coloca debajo del ícono. El contenido del botón está centrado.	
Centrado	El texto del ícono está centrado vertical y horizontalmente en el botón. Este parámetro es útil, por ejemplo, para el texto incluido en un ícono.	

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
textPlacement	cadena	"left", "top", "right", "bottom", "center"

## Objetos soportados

[Botón](#) (todos los estilos excepto [Ayuda](#)) - [Casilla de selección](#) - [Botón radio](#)

## Margen vertical

Esta propiedad permite definir el tamaño (en píxeles) de los márgenes verticales del botón. Este margen delimita el área que el ícono del botón y el título no deben sobrepasar.

Este parámetro es útil, por ejemplo, cuando la imagen de fondo contiene bordes.

Esta propiedad funciona junto con la propiedad [Margen horizontal](#).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
customBorderY	number	Para usar con el estilo "personalizado". Mínimo: 0

## Objetos soportados

[Botón personalizado](#) - [Casilla de selección personalizada](#) - [Botón radio personalizado](#)

## Con menú pop-up

Esta propiedad permite mostrar un símbolo que aparece como un triángulo en el botón para indicar la presencia de un menú emergente adjunto:



La apariencia y ubicación de este símbolo depende del estilo del botón y de la plataforma actual.

## Vinculados y separados

Para asociar un símbolo de menú emergente a un botón, hay dos opciones de visualización disponibles:

Enlazado	Separado
Toolbar	Toolbar

La disponibilidad efectiva de un modo "separado" depende del estilo del botón y de la plataforma.

Cada opción precisa la relación entre el botón y el menú emergente asociado:

- Cuando el menú emergente está **\*\*separado\*\***, al hacer clic en la parte izquierda del botón se ejecuta directamente la acción actual del botón; esta acción puede modificarse mediante el menú emergente accesible en la parte derecha del botón.
- Cuando el menú emergente está **\*\*vinculado\*\***, un simple clic en el botón sólo muestra el menú emergente. Sólo la selección de la acción en el menú emergente provoca su ejecución.

## Gestión del menú emergente

Es importante señalar que la propiedad "Con menú emergente" sólo gestiona el aspecto gráfico del botón. La visualización del menú emergente y sus valores deben ser manejados enteramente por el desarrollador, particularmente utilizando `eventos formulario` y los comandos [Menú emergente dinámico](#) y [Menú emergente](#).

## Gramática JSON

Nombre	Tipos de datos	Valores posibles
popupPlacement	cadena	<ul style="list-style-type: none"><li>◦ "none"</li><li>◦ "linked"</li><li>◦ "separated"</li></ul>

## Objetos soportados

[Botón barra de herramientas](#) - [Botón bisel](#) - [Botón de bisel redondeado](#) - [Botón OS X Gradient](#) - [Botón OS X Textured](#) - [Botón Office XP](#) - [Botón círculo](#) - [Personalizado](#)

# Área Web

---

## Acceder a los métodos 4D

Puede llamar a los métodos 4D desde el código JavaScript ejecutado en un área web y recibir valores a cambio. Para poder llamar a los métodos 4D desde un área Web, debe activar la propiedad de accesibilidad de los métodos 4D ("todos").

Esta propiedad sólo está disponible si el área web [utiliza el motor de renderizado web integrado](#).

Cuando esta propiedad está activada, se instancia un objeto JavaScript especial llamado `$4d` en el área web, que puede [utilizar para gestionar las llamadas a los métodos proyecto de 4D](#).

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
methodsAccessibility	cadena	"none" (por defecto), "all"

### Objetos soportados

#### Área Web

---

## Variable Progression

Nombre de una variable de tipo Longint. Esta variable recibirá un valor entre 0 y 100, que representa el porcentaje de finalización de la carga de la página en el área web. Actualizado automáticamente por 4D, no puede ser modificado manualmente.

As of 4D v19 R5, this variable is no longer updated in Web Areas using the [Windows system rendering engine](#).

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
progressSource	cadena	Nombre de una variable Longint

### Objetos soportados

#### Área Web

---

## URL

La variable URL es de tipo cadena. Contiene la URL cargada o que está siendo cargada por el área web asociada. La asociación entre la variable y el área web funciona en ambas direcciones:

- Si el usuario asigna una nueva URL a la variable, esta URL es cargada automáticamente por el área web.
- Toda la navegación que se realice dentro del área web actualizará automáticamente el contenido de la variable.

Esquemáticamente, esta variable funciona como el área de direcciones de un navegador web. Puede representarlo a

través de un área de texto sobre el área Web.

## Variable URL y comando WA OPEN URL

La variable URL produce los mismos efectos que el comando [WA OPEN URL](#). No obstante, hay que señalar las siguientes diferencias:

- Para el acceso a los documentos, esta variable sólo acepta URLs que cumplan con el RFC ("file://c:/My%20Doc") y no los nombres de ruta del sistema ("c:¥MyDoc"). El comando [WA OPEN URL](#) acepta ambas notaciones.
- Si la variable URL contiene una cadena vacía, el área web no intenta cargar la URL. El comando [WA OPEN URL](#) genera un error en este caso.
- Si la variable URL no contiene un protocolo (http, mailto, archivo, etc.), el área web añade "http://", lo que no ocurre con el comando [WA OPEN URL](#).
- Cuando el área web no se muestra en el formulario (cuando se encuentra en otra página del formulario), la ejecución del comando [WA OPEN URL](#) no tiene ningún efecto, mientras que la asignación de un valor a la variable URL puede utilizarse para actualizar la URL actual.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
urlSource	cadena	Una URL.

### Objetos soportados

#### Área Web

## Utilizar el motor de renderizado web integrado

Esta opción permite elegir entre dos motores de renderizado para el área web, dependiendo de las particularidades de su aplicación:

- no marcado - `valor JSON: sistema` (por defecto): en este caso, 4D utiliza el "mejor" motor correspondiente al sistema. Esto significa que usted se beneficia automáticamente de los últimos avances en la renderización web, a través de HTML5 o JavaScript. Sin embargo, es posible que note algunas diferencias de renderizado entre plataformas. En Windows, 4D utiliza Microsoft Edge WebView2. En macOS, 4D utiliza la versión actual de WebKit (Safari).

On Windows, if Microsoft Edge WebView2 is not installed, 4D uses the embedded engine as system rendering engine. To know if it is installed in your system, look for "Microsoft Edge WebView2 Runtime" in your applications panel.

- `checked - JSON value: embedded` : In this case, 4D uses the Chromium Embedded Framework (CEF). La utilización del motor web integrado significa que la representación de las áreas web y su funcionamiento en su aplicación son idénticos independientemente de la plataforma utilizada para ejecutar 4D (no obstante, pueden observarse ligeras variaciones de píxeles o diferencias relacionadas con la implementación de la red). La utilización del motor web integrado significa que la representación de las áreas web y su funcionamiento en su aplicación son idénticos independientemente de la plataforma utilizada para ejecutar 4D (no obstante, pueden observarse ligeras variaciones de píxeles o diferencias relacionadas con la implementación de la red).

El motor CEF tiene las siguientes limitaciones:

- [WA SET PAGE CONTENT](#): using this command requires that at least one page is already loaded in the area (through a call to [WA OPEN URL](#) or an assignment to the URL variable associated to the area).
- Cuando se activa soltar URLs mediante el selector `WA enable URL drop` del comando [WA SET PREFERENCE](#), la primera soltada debe ir precedida de al menos una llamada a [WA OPEN URL](#) o una asignación a la variable URL asociada al área.

### Gramática JSON

Nombre	Tipos de datos	Valores posibles
webEngine	cadena	"embedded", "system"

Objetos soportados

[Área Web](#)

# On Activate

Code	Puede ser llamado por	Definición
11	Formulario	La ventana del formulario se convierte en la ventana que se encuentra más adelante o el contenedor del subformulario obtiene el foco

## Descripción

Si la ventana de un formulario fue enviada al fondo, este evento es llamado cuando la ventana se convierte en la ventana activa.

Este evento se aplica al formulario en su conjunto y no a un objeto en particular. Por lo tanto, si se selecciona la propiedad de evento formulario `On Activate`, sólo se llamará al método formulario.

En el caso de un subformulario, este evento se pasa al subformulario cuando el contenedor obtiene el foco (si tiene la propiedad [focusable](#)).

# On After Edit

Code	Puede ser llamado por	Definición
45	Área 4D View Pro - Área 4D Write Pro - Combo Box - Formulario - Entrada - Lista jerárquica - List Box - Columna List Box	El contenido del objeto introducible que tiene el foco acaba de ser modificado

## Descripción

### Caso general

Este evento se puede utilizar para filtrar la entrada de datos en los objetos editables por teclado en el nivel más bajo.

Cuando se utiliza, este evento se genera después de cada cambio realizado en el contenido de un objeto editable, independientemente de la acción que haya provocado la modificación, es decir:

- Acciones de edición estándar que modifican el contenido como pegar, cortar, borrar o cancelar;
- Soltar un valor (acción similar a pegar);
- Toda entrada de teclado realizada por el usuario; en este caso, el evento `On After Edit` se genera después de los eventos `On Before Keystroke` y `On After Keystroke`, si se utilizan.
- Cualquier modificación realizada mediante un comando del lenguaje que simule una acción del usuario (es decir, `POST KEY`).

Within the `On After Edit` event, text data being entered is returned by the `Get edited text` command.

## 4D View Pro

El objeto devuelto por el comando `FORM Event` contiene:

Propiedad	Tipo	Descripción
code	entero largo	On After Edit
description	texto	"On After Edit"
objectName	texto	Nombre del área 4D View Pro
sheetName	texto	Nombre de la hoja del evento
action	texto	"editChange", "valueChanged", "DragDropBlock", "DragFillBlock", "formulaChanged", "clipboardPasted"

En función del valor de la propiedad `action`, el `objeto evento` contendrá propiedades adicionales.

`action = editChange`

Propiedad	Tipo	Descripción
range	objeto	Rango de celdas
editingText	variant	El valor proveniente del editor actual

`action = valueChanged`

Propiedad	Tipo	Descripción
range	objeto	Rango de celdas
oldValue	variant	Valor de la celda antes de la modificación
newValue	variant	Valor de la celda luego de la modificación

action = DragDropBlock

Propiedad	Tipo	Descripción
fromRange	objeto	Rango de celdas fuente (que se arrastra)
toRange	objeto	Rango de la celda de destino (ubicación de soltar)
copy	booleano	Indica si el rango fuente se copia o no
insert	booleano	Indica si el rango fuente se inserta o no

action = DragFillBlock

Propiedad	Tipo	Descripción	fillDirection	longint	Dirección del relleno.
fillRange	objeto	Gama utilizada para el relleno			
autoFillType	longint	Valor utilizado para el relleno. <ul style="list-style-type: none"> <li>• 0: las celdas se llenan con todos los datos (valores, formato y fórmulas)</li> <li>• 1: las celdas se llenan con datos automáticamente secuenciales</li> <li>• 2: Las celdas se llenan sólo con el formato</li> <li>• 3: Las celdas se llenan de valores pero sin formato</li> <li>• 4: Se eliminan los valores de las celdas</li> <li>• 5: Las celdas se llenan automáticamente.</li> </ul>			<ul style="list-style-type: none"> <li>• 0: Se llenan las celdas de la izquierda</li> <li>• 1: Se llenan las celdas a la derecha</li> <li>• 2: Las celdas de arriba se llenan</li> <li>• 3: Las celdas de abajo se llenan</li> </ul>

action = formulaChanged

Propiedad	Tipo	Descripción
range	objeto	Rango de celdas
formula	texto	La fórmula introducida

action = clipboardPasted

Propiedad	Tipo	Descripción
range	objeto	Rango de celdas
pasteOption	entero largo	Indica lo que se pega desde el portapapeles: <ul style="list-style-type: none"> <li>• 0: se pega todo (valores, formato y fórmulas)</li> <li>• 1: solo se pegan los valores</li> <li>• 2: sólo se pega el formato</li> <li>• 3: solo se pegan las fórmulas</li> <li>• 4: los valores y el formato se pegan (no las fórmulas)</li> <li>• 5: las fórmulas y el formato se pegan (no los valores)</li> </ul>
pasteData	objeto	Los datos del portapapeles a pegar <ul style="list-style-type: none"> <li>• "text" (texto): el texto del portapapeles</li> <li>• "html" (texto): el código HTML del portapapeles</li> </ul>

## Ejemplo

Aquí hay un ejemplo de manejo de un evento `On After Edit`:

```
If(FORM Event.code=On After Edit)
  If(FORM Event.action="valueChanged")
    ALERT("WARNING: You are currently changing the value\
      from "+String(FORM Event.oldValue)+\
      " to "+String(FORM Event.newValue)+"!")
  End if
End if
```

El ejemplo anterior podría generar un objeto evento como este:

```
{
  "code":45,
  "description":"On After Edit",
  "objectName":"ViewProArea",
  "sheetname":"Sheet1",
  "action":"valueChanged",
  "range": {area:ViewProArea,ranges:[{column:1,row:2,sheet:1}]},
  "oldValue":"The quick brown fox";
  "newValue":"jumped over the lazy dog";
}
```

# On After Keystroke

Code	Puede ser llamado por	Definición
28	<a href="#">4D Write Pro area</a> - <a href="#">Combo Box</a> - <a href="#">Form</a> - <a href="#">Input</a> - <a href="#">List Box</a> - <a href="#">List Box Column</a>	Un personaje está a punto de ser introducido en el objeto que tiene el foco. <code>Get edited text</code> devuelve el texto del objeto incluyendo este carácter.

► Histórico

## Descripción

El evento `On After Keystroke` generalmente puede ser reemplazado por el evento `On After Edit` (ver abajo).

Después de que las propiedades de evento [`On Before Keystroke`] (`onBeforeKeystroke.md`) y `On After Keystroke` estén seleccionadas para un objeto, puede detectar y manejar las presiones de las teclas dentro del objeto, utilizando el comando `FORM event` que devolverá `On Before Keystroke` y luego `On After Keystroke` (para más información, consulte la descripción del comando `Get edited text`).

Estos eventos también son activados por comandos del lenguaje que simulan una acción del usuario como `POST KEY`.

El evento `On After Keystroke` no se genera:

- en el método [de las columnas de list box](#) excepto cuando se está editando una celda (sin embargo se genera en cualquier caso en el método de [list box](#)),
- cuando las modificaciones usuario no se realizan con el teclado (pegar, arrastrar y soltar, casilla de verificación, lista desplegable, combo box). Para procesar estos eventos, debe utilizar [On After Edit](#).

## Secuencia de tecla

Cuando una entrada requiere una secuencia de presiones de teclas, los eventos `On Before Keystroke` y [`On After Keystroke`] se generan sólo cuando el usuario valida completamente la entrada. El comando `Keystroke` devuelve el carácter validado. Este caso se da principalmente:

- cuando se utilizan las teclas "muertas" como ^ o ~: los eventos se generan sólo cuando se introduce el carácter extendido eventualmente (por ejemplo, "ê" o ñ),
- cuando un IME (editor de métodos de entrada) muestra una caja de diálogo intermedia en la que el usuario puede introducir una combinación de caracteres: los eventos se generan sólo cuando el diálogo IME se valida.

## Ver también

[On Before Keystroke](#).

# On After Sort

Code	Puede ser llamado por	Definición
30	<a href="#">List Box - Columna de List Box</a>	Se acaba de realizar una ordenación estándar en una columna del list box.

## Descripción

Este evento se genera justo después de realizar una ordenación estándar ( *es decir*, NO se genera si \$0 devuelve -1 en el evento [On Header Click](#) ). Este mecanismo es útil para almacenar las direcciones de la última ordenación realizada por el usuario. En este caso, el comando `Self` devuelve un puntero a la variable del encabezado de la columna ordenada.

# On Alternative Click

Code	Puede ser llamado por	Definición
38	<a href="#">Botón</a> - <a href="#">List Box</a> - <a href="#">Columna de List Box</a>	<ul style="list-style-type: none"><li>• Botones: el área "flecha" de un botón se presiona</li><li>• List box: en una columna de un array, se hace clic en un botón de selección (atributo "alternateButton")</li></ul>

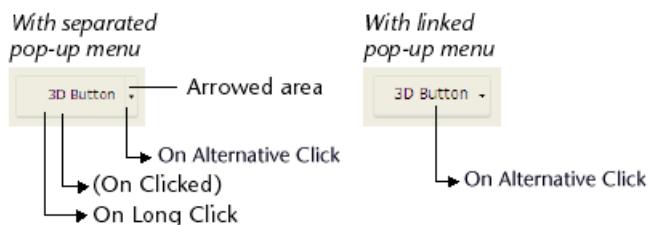
## Descripción

### Botones

Algunos estilos de botón pueden ser [vinculados a un menú emergente](#) y mostrar un triángulo. Al hacer clic en este triángulo, aparece una ventana emergente de selección que brinda un conjunto de acciones alternativas en relación con la acción del botón principal.

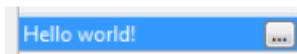
4D permite gestionar este tipo de botones utilizando el evento `On Alternative Click`. Este evento se genera cuando el usuario hace clic en el triángulo (en cuanto se mantiene presionado el botón del ratón):

- Si el menú emergente está separado, el evento sólo se genera cuando se hace clic en la parte del botón con la flecha.
- Si el menú emergente está asociado, el evento se genera cuando se hace clic en cualquier parte del botón. Tenga en cuenta que el evento `On Long Click` no se puede generar con este tipo de botón.



### List box

Este evento se genera en las columnas de [list box de tipo array objeto](#), cuando el usuario hace clic en un botón de selección de widget (atributo "alternateButton").



Ver la [descripción del atributo "alternateButton"](#).

# On Before Data Entry

Code	Puede ser llamado por	Definición
41	<a href="#">List Box - Columna de List Box</a>	Una celda de list box está a punto de cambiar al modo de edición

## Descripción

Este evento se genera justo antes de que se edite una celda del list box (antes de que se muestre el cursor de entrada). Este evento permite al desarrollador, por ejemplo, mostrar un texto diferente dependiendo de si el usuario está en el modo de visualización o de edición.

Cuando el cursor llega a la celda, se genera el evento `On Before Data Entry` en el list box o método de la columna.

- Si, en el contexto de este evento, \$0 se define como -1, la celda se considera como no editable. Si el evento se generó después de presionar Tab o Mayús+Tab, el foco pasa a la siguiente celda o a la anterior, respectivamente.
- Si \$0 no es -1 (por defecto \$0 es 0), la celda se puede introducir y pasa al modo de edición.

Ver también la sección [Gestión de entradas](#).

# On Before Keystroke

Code	Puede ser llamado por	Definición
17	<a href="#">4D Write Pro area</a> - <a href="#">Combo Box</a> - <a href="#">Form</a> - <a href="#">Input</a> - <a href="#">List Box</a> - <a href="#">List Box Column</a>	Un personaje está a punto de ser introducido en el objeto que tiene el foco. <code>Get edited text</code> devuelve el texto del objeto sin este carácter.

► Histórico

## Descripción

Después de haber seleccionado los eventos `On Before Keystroke` y `On After Keystroke event` para un objeto, puede detectar y manejar las presiones de las teclas dentro del objeto, utilizando el comando `Form event code` que devolverá `On Before Keystroke` y luego `On After Keystroke event` (para más información, consulte la descripción del comando `Get edited text`). En el evento `On Before Keystroke`, se puede utilizar el comando `FILTER KEYSTROKE` para filtrar los caracteres digitados.

Estos eventos también son activados por comandos del lenguaje que simulan una acción del usuario como `POST KEY`.

El evento `On Before Keystroke` no se genera:

- en un método `columnas de list box` excepto cuando se está editando una celda (sin embargo se genera en cualquier caso en el método de `list box`),
- cuando las modificaciones usuario no se realizan con el teclado (pegar, arrastrar y soltar, casilla de verificación, lista desplegable, combo box). Para procesar estos eventos, debe utilizar `On After Edit`.

## Objetos no editables

El evento `On Before Keystroke` puede generarse en objetos no introducibles, por ejemplo, en un list box aunque las celdas del list box no sean introducibles, o las líneas no sean seleccionables. Esto permite crear interfaces en las que el usuario puede desplazarse dinámicamente a una línea específica en un list box introduciendo las primeras letras de un valor. En el caso de que las celdas del cuadro del list box sean editables, puede utilizar el comando `Is editing text` para saber si el usuario está realmente introduciendo texto en una celda o está utilizando la función de tecleo predictivo y entonces, ejecutar el código apropiado.

## Secuencia de tecla

Cuando una entrada requiere una secuencia de presiones de teclas, los eventos `On Before Keystroke` y `On After Keystroke` se generan sólo cuando el usuario valida completamente la entrada. El comando `Keystroke` devuelve el carácter validado. Este caso se da principalmente:

- cuando se utilizan las teclas "muertas" como ^ o ~: los eventos se generan sólo cuando se introduce el carácter extendido eventualmente (por ejemplo, "ê" o ñ),
- cuando un IME (editor de métodos de entrada) muestra una caja de diálogo intermedia en la que el usuario puede introducir una combinación de caracteres: los eventos se generan sólo cuando el diálogo IME se valida.

## Ver también

[On After Keystroke](#).

# On Begin Drag Over

Code	Puede ser llamado por	Definición
17	Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Lista desplegable - Formulario - Lista jerárquica - Área de entrada - List Box - Columna List Box - Botón imagen - Pop up menu image - Área de plug-in - Indicador de progreso - Botón radio - Regla - Spinner - Splitter - Stepper - Pestaña	Se está arrastrando un objeto

## Descripción

El evento de formulario `On Begin Drag Over` puede ser seleccionado para todos los objetos formulario que puedan ser arrastrados. Se genera en todos los casos en que el objeto tiene la propiedad **Draggable**. Se puede llamar desde el método del objeto fuente o desde el método formulario del objeto fuente.

A diferencia del evento de formulario `On Drag Over`, `On Begin Drag Over` se llama dentro del contexto del objeto fuente de la acción de arrastrar.

El evento `On Begin Drag Over` es útil para preparar la acción de arrastrar. Puede utilizarse para:

- Añadir los datos y las firmas al portapapeles (vía el comando `APPEND DATA TO PASTEBOARD`).
- Utilizar un ícono personalizado durante la acción de arrastre (vía el comando `SET DRAG ICON`).
- Aceptar o rechazar el arrastre vía \$0 en el método del objeto arrastrado.
  - Para indicar que se aceptan las acciones de arrastre, el método del objeto fuente debe devolver 0 (cero); por tanto, debe ejecutar `$0:=0`.
  - Para indicar que se rechazan las acciones de arrastre, el método del objeto fuente debe devolver -1 (menos uno); por tanto, debe ejecutar `$0:=-1`.
  - Si no se devuelve ningún resultado, 4D considera que las acciones de arrastre son aceptadas.

Los datos 4D se colocan en el portapapeles antes de llamar al evento. Por ejemplo, en el caso de arrastrar sin la acción Arrastre automático, el texto arrastrado ya está en portapapeles cuando se llama al evento.

# On Begin URL Loading

Code	Puede ser llamado por	Definición
47	<a href="#">Área Web</a>	Se carga una nueva URL en el área web

## Descripción

Este evento se genera al inicio de la carga de una nueva URL en el área web. La variable `URL` asociada al área web puede utilizarse para conocer la URL que se está cargando.

La URL que se está cargando es diferente de la [URL actual](#) (consulte la descripción del comando `WA Get current URL`).

# On Bound Variable Change

Code	Puede ser llamado por	Definición
54	Formulario	La variable vinculada a un subformulario se modifica

## Descripción

Este evento se genera en el contexto del método formulario de un [subformulario](#) en cuanto se asigna un valor a la variable vinculada con el subformulario del formulario padre (incluso si se reasigna el mismo valor) y si el subformulario pertenece a la página actual del formulario o a la página 0.

Para más información, consulte la sección [Gestión de la variable vinculada](#).

# On Clicked

Code	Puede ser llamado por	Definición
4	Área 4D View Pro - Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Combo Box - Lista desplegable - Formulario - Lista jerárquica - Entrada - List Box - Columna de List Box - Botón imagen - Pop up menu imagen - Área de plug-in - Indicador de progreso - Botón radio - Regla - Spinner - Splitter - Stepper - Pestaña	Se ha producido un clic en un objeto

## Descripción

El evento `On Clicked` se genera cuando el usuario hace clic en un objeto.

Ciertos objetos de formulario pueden activarse con el teclado. Por ejemplo, una vez que una casilla de selección recibe el foco, se puede introducir con la barra espaciadora. En tal caso, se sigue generando el evento `On Clicked`.

El evento `On Clicked` suele producirse una vez que se suelta el botón del ratón. Sin embargo, hay varias excepciones:

- **Botones invisibles:** el evento `On Clicked` se produce en cuanto se hace clic y no espera a que se suelte el botón del ratón.
- **Reglas:** si la opción de **método de ejecución del objeto** se define en `true`, el evento `On Clicked` se produce en cuanto se hace clic.
- **Combo box:** el evento `On Clicked` ocurre sólo si el usuario selecciona otro valor en el menú asociado. Un **combo box** debe ser tratado como un área de texto introducible cuya lista desplegable asociada ofrece valores por defecto. Por lo tanto, se maneja la entrada de datos dentro de un combo box a través de los eventos `On Before Keystroke`, `On After Keystroke` y `On Data Change`.
- **Listas desplegables:** el evento `On Clicked` ocurre sólo si el usuario selecciona otro valor en el menú. El evento `On Data Change` permite detectar la activación del objeto cuando se selecciona un valor diferente al actual
- Cuando una celda de entrada del list box está **siendo editada**, se genera el evento `On Clicked` cuando se presiona el botón del ratón, permitiendo utilizar el comando `Contextual click` por ejemplo.

En el contexto de un evento `On Clicked`, se puede comprobar el número de clics realizados por el usuario utilizando el comando `Clickcount`.

## On Clicked y On Double Clicked

Una vez que la propiedad de evento de objeto `On Clicked` o `On Double Clicked` es seleccionada para un objeto, puede detectar y controlar los clics dentro o sobre el objeto utilizando el comando `FORM event` que devuelve `On Clicked` o `On Double Clicked`, según el caso.

Si se seleccionan ambos eventos para un objeto, se generará el evento `On Clicked` y luego el evento `On Double Clicked` cuando el usuario haga doble clic en el objeto.

## 4D View Pro

Este evento se genera cuando el usuario hace clic en cualquier parte en un documento 4D View Pro. En este contexto, el **objeto evento** devuelto por el comando `FORM Event` contiene:

Propiedad	Tipo	Descripción
code	entero largo	On Clicked
description	texto	"On Clicked"
objectName	texto	Nombre del área 4D View Pro
sheetName	texto	Nombre de la hoja del evento
range	objeto	Rango de celdas

## Ejemplo

```
If(FORM Event.code=On Clicked)
    VP SET CELL STYLE(FORM Event.range;New object("backColor";"green"))
End if
```

# On Close Box

Code	Puede ser llamado por	Definición
22	Formulario	Se ha presionado la caja de cierre de la ventana

## Descripción

El evento `On Close Box` se genera cuando el usuario hace clic en una caja de cierre de la ventana.

## Ejemplo

Este ejemplo muestra cómo responder a un evento de cierre de ventana con un formulario utilizado para la entrada de datos de registro:

```
//Método para un formulario de entrada
$vpFormTable:=Current form table
Case of
//...
:(Form event code=On Close Box)
If(Modified record($vpFormTable->))
    CONFIRM("Este registro ha sido modificado. Save Changes?")
    If(OK=1)
        ACCEPT
    Else
        CANCEL
    End if
Else
    CANCEL
End if
//...
End case
```

# On Close Detail

Code	Puede ser llamado por	Definición
26	Formulario - <a href="#">List Box</a>	Ha dejado el formulario detallado y está volviendo al formulario de salida

## Descripción

El evento `On Close Detail` puede utilizarse en los siguientes contextos:

- Formularios de salida: el formulario detallado se cierra y el usuario vuelve al formulario listado. Este evento no se puede seleccionar para los formularios proyecto, sólo está disponible con los formularios tabla.
- List box [de tipo selección](#): este evento se genera cuando un registro mostrado en el [formulario detallado](#) asociado a un list box de tipo selección está a punto de cerrarse (independientemente de que el registro se haya modificado o no).

# On Collapse

Code	Puede ser llamado por	Definición
44	<a href="#">Lista jerárquica - List Box</a>	Un elemento de la lista jerárquica o del list box jerárquico se ha contraído por medio de un clic o una presión de tecla

## Descripción

- [Lista jerárquica](#): este evento se genera cada vez que se colapsa un elemento de la lista jerárquica con un clic del ratón o una pulsación del teclado.
- [List box jerárquicos](#): este evento se genera cuando se colapsa una línea del list box jerárquico.

## Ver también

[On Expand](#)

# On Column Moved

Code	Puede ser llamado por	Definición
32	<a href="#">List Box - Columna de List Box</a>	Una columna de list box es movida por el usuario por medio de arrastrar y soltar

## Descripción

Este evento se genera cuando una columna de list box es movida por el usuario utilizando la función de arrastrar y soltar ([si se permite](#)). No se genera si la columna se arrastra y luego se suelta en su ubicación inicial.

El comando `LISTBOX MOVED COLUMN NUMBER` devuelve la nueva posición de la columna.

# On Column Resize

Code	Puede ser llamado por	Definición
33	Área 4D View Pro - List Box - Columna de List Box	El ancho de una columna es modificado directamente por el usuario o en consecuencia de un redimensionamiento de la ventana del formulario

## Descripción

### List Box

Este evento se genera cuando el ancho de una columna en el list box es modificado por un usuario. El evento se activa "en directo", es decir, se envía continuamente durante el evento, mientras se redimensiona el list box o la columna en cuestión. Este redimensionamiento es realizado manualmente por un usuario, o puede ocurrir como resultado de que el list box y su(s) columna(s) sean redimensionados junto con la propia ventana del formulario (ya sea que el formulario sea redimensionado manualmente o utilizando el comando `RESIZE FORM WINDOW`).

El evento `On Column Resize` no se activa cuando se redimensiona una falsa columna.

### 4D View Pro

Este evento se genera cuando el ancho de una columna es modificado por un usuario. En este contexto, el `objeto evento` devuelto por el comando `F0RM Event` contiene:

Propiedad	Tipo	Descripción
code	entero largo	On Column Resize
description	texto	"On Column Resize"
objectName	texto	Nombre del área 4D View Pro
sheetName	texto	Nombre de la hoja del evento
range	objeto	Rango de celdas de las columnas cuyo ancho ha cambiado
header	booleano	True si la columna de encabezado de línea (primera columna) se redimensiona, si no, false

### Ejemplo

```
If(FORM Event.code=On Column Resize)
  VP SET CELL STYLE(FORM Event.range;New object("hAlign";vk horizontal align right))
End if
```

# On Data Change

Code	Puede ser llamado por	Definición
20	Área 4D Write Pro - Lista desplegable - Formulario - Lista jerárquica - Área de entrada - List Box - Columna de List Box - Área de Plug-in - Indicador de progresión - Regla - Spinner - Stepper - Sub-formulario	Se han modificado los datos de un objeto

## Descripción

Cuando se selecciona la propiedad de evento `On Data Change` para un objeto, se puede detectar y manejar el cambio del valor de la fuente de datos, utilizando el comando `FORM Event`.

El evento se genera en cuanto la variable asociada al objeto es actualizada internamente por 4D (es decir, en general, cuando la zona de entrada del objeto pierde el foco).

Con los [subformularios](#), el evento `On Data Change` se dispara cuando el valor de la variable del objeto subformulario ha sido modificado.

# On Deactivate

Code	Puede ser llamado por	Definición
12	Formulario	La ventana del formulario deja de ser la ventana activa

## Descripción

Si la ventana de un formulario era la ventana del primer plano, este evento es llamado cuando la ventana es enviada al fondo.

Este evento se aplica al formulario en su conjunto y no a un objeto en particular. Por lo tanto, si se selecciona la propiedad de evento formulario `On Deactivate`, sólo se llamará al método formulario.

## Ver también

[On Activate](#)

# On Delete Action

Code	Puede ser llamado por	Definición
58	<a href="#">Lista jerárquica - List Box</a>	El usuario intenta eliminar un elemento

## Descripción

Este evento se genera cada vez que un usuario intenta eliminar los elementos seleccionados presionando una tecla de eliminación (Borrar o Retroceso) o seleccionando un elemento de menú cuya acción estándar asociada es "Borrar" (como el comando Borrar del menú Edición).

Tenga en cuenta que la generación del evento es la única acción que realiza 4D: el programa no borra ningún elemento. Es el desarrollador quien debe gestionar el borrado y los mensajes de advertencia previos que se muestren.

# On Display Detail

Code	Puede ser llamado por	Definición
8	Formulario - List Box	Un registro está a punto de ser mostrado en un formulario lista o una línea está a punto de ser mostrada en un list box.

## Descripción

El evento `On Display Detail` puede utilizarse en los siguientes contextos:

### Formulario de salida

Un registro está a punto de ser visualizado en un formulario de lista que se muestra vía `DISPLAY SELECTION` y `MODIFY SELECTION`.

Este evento no se puede seleccionar para los formularios proyecto, sólo está disponible con los formularios tabla.

En este contexto, se desencadena la siguiente secuencia de llamadas a métodos y eventos de formulario:

- Para cada registro:
  - Para cada objeto en el área detallada:
    - Método objeto con el evento `On Display Detail`
  - Método formulario con el evento `On Display Detail`

El área del encabezado se maneja con el evento `On Header`.

Llamar a un comando 4D que muestra una caja de diálogo desde el evento `On Display Detail` no está permitido y provocará un error de sintaxis. Más concretamente, los comandos en cuestión son: `ALERT`, `DIALOG`, `CONFIRM`, `Request`, `ADD RECORD`, `MODIFY RECORD`, `DISPLAY SELECTION` y `MODIFY SELECTION`.

### List box selección

Este evento se genera cuando se muestra una línea de list box [de tipo selección](#).

### Número de línea mostrado

El comando 4D `Número de línea mostrado` funciona con el evento formulario `On Display Detail`. Devuelve el número de la línea que se está procesando mientras se visualiza en pantalla una lista de registros o de líneas de list box.

# On Double Clicked

Code	Puede ser llamado por	Definición
13	Área 4D View Pro - Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Combo Box - Lista desplegable - Formulario - Lista jerárquica - Entrada - List Box - Columna de List Box - Botón imagen - Pop up menu imagen - Área de plug-in - Indicador de progreso - Botón radio - Regla - Spinner - Splitter - Stepper - Pestaña	Se ha efectuado un doble clic en un objeto

## Descripción

El evento `On Double Clicked` se genera cuando el usuario hace doble clic en un objeto. El tiempo máximo de separación de un doble clic se define en las preferencias del sistema.

Si la propiedad `On Clicked` o `On Double Clicked` de evento de objeto de `onDoubleClicked.md` está seleccionada para un objeto, puede detectar y manejar los clics dentro o sobre el objeto, utilizando el comando `FORM event` que devuelve `On Clicked` o `On Double Clicked`, dependiendo del caso.

Si se seleccionan ambos eventos para un objeto, se generará el evento `On Clicked` y luego el evento `On Double Clicked` cuando el usuario haga doble clic en el objeto.

## 4D View Pro

Este evento se genera cuando el usuario hace doble clic en cualquier parte en un documento 4D View Pro. En este contexto, el `objeto evento` devuelto por el comando `FORM Event` contiene:

Propiedad	Tipo	Descripción
code	entero largo	13
description	texto	"On Double Clicked"
objectName	texto	Nombre del área 4D View Pro
sheetName	texto	Nombre de la hoja del evento
range	objeto	Rango de celdas

## Ejemplo

```
If(FORM Event.code=On Double Clicked)
  $value:=VP Get value(FORM Event.range)
End if
```

# On Drag Over

Code	Puede ser llamado por	Definición
21	Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Lista desplegable - Lista jerárquica - Área de entrada -List Box - Columna de List Box - Botón imagen - Imagen del menú emergente - Área de plug-in - Indicador de progreso - Botón radio - Regla -Spinner - Splitter - Stepper - Pestaña	Los datos se pueden soltar en un objeto

## Descripción

El evento `On Drag Over` se envía repetidamente al objeto de destino cuando el puntero del ratón se mueve sobre el objeto. Normalmente, en respuesta a este evento:

- Obtenga los datos y las firmas que se encuentran en portapapeles (mediante el comando `GET PASTEBORD DATA` ).
- Según la naturaleza y el tipo de datos en el portapapeles, se acepta o rechaza el arrastrar y soltar.

Para aceptar el arrastre, el método del objeto destino debe devolver 0 (cero), por lo que se escribe `$0:=0` . Para rechazar el arrastre, el método del objeto debe devolver -1 (menos uno), por lo que se escribe `$0:=-1` . Durante un evento `On Drag Over` , 4D trata el método objeto como una función. Si no se devuelve ningún resultado, 4D asume que el arrastre es aceptado.

Si acepta el arrastre, el objeto de destino se resalta. Si rechaza el arrastre, el destino no se resalta. Aceptar el arrastre no significa que los datos arrastrados vayan a ser insertados en el objeto de destino. Esto sólo significa que si se soltara el botón del ratón en este punto, el objeto de destino aceptaría los datos arrastrados y se dispararía el evento `On Drop` .

Si no se procesa el evento `On Drag Over` para un objeto soltable, ese objeto será resaltado para todas las operaciones de arrastre, sin importar la naturaleza y el tipo de los datos arrastrados.

El evento `On Drag Over` es el medio por el que se controla la primera fase de una operación de arrastrar y soltar. No sólo puede probar si los datos arrastrados son de un tipo compatible con el objeto de destino, y luego aceptar o rechazar el arrastre; puede notificar simultáneamente al usuario de este hecho, porque 4D resalta (o no) el objeto de destino, basándose en su decisión.

El código que maneja un evento `On Drag Over` debe ser corto y ejecutarse rápidamente, porque ese evento se envía repetidamente al objeto de destino actual, debido a los movimientos del ratón.

## Ver también

[On Begin Drag Over](#)

# On Drop

Code	Puede ser llamado por	Definición
16	Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Lista desplegable - Formulario - Lista jerárquica - Área de entrada - List Box - Columna List Box - Botón imagen - Pop up menu image - Área de plug-in - Indicador de progreso - Botón radio - Regla - Spinner - Splitter - Stepper - Pestaña	Los datos se han depositado en un objeto

## Descripción

El evento `On Drop` se envía una vez al objeto de destino cuando el puntero del ratón se mueve sobre el objeto. Este evento es la segunda fase de la operación de arrastrar y soltar, en la que se realiza una operación en respuesta a la acción del usuario.

Este evento no se envía al objeto si el arrastre no fue aceptado durante los eventos `On Drag Over`. Si se procesa el evento `On Drag Over` para un objeto y se rechaza un arrastre, no se produce el evento `On Drop`. Así, si durante el evento `On Drag Over` se ha probado la compatibilidad de los tipos de datos entre los objetos origen y destino y se ha aceptado una posible caída, no es necesario volver a comprobar los datos durante el evento `On Drop`. Ya sabe que los datos son adecuados para el objeto de destino.

Ver también

[On Begin Drag Over](#)

# On End URL Loading

Code	Puede ser llamado por	Definición
49	<a href="#">Área Web</a>	Se han cargado todos los recursos de la URL

## Descripción

Este evento se genera una vez que se ha completado la carga de todos los recursos de la URL. Puedes llamar al comando `WA Get current URL` para conocer la URL que se cargó.

# On Expand

Code	Puede ser llamado por	Definición
44	<a href="#">Lista jerárquica - List Box</a>	Un elemento de la lista jerárquica o del list box jerárquico se ha expandido por medio de un clic o una tecla

## Descripción

- [Lista jerárquica](#): este evento se genera cada vez que se expande un elemento de la lista jerárquica con un clic del ratón o una pulsación del teclado.
- [List box jerárquicos](#): este evento se genera cuando se expande una línea del list box jerárquico.

## Ver también

[On Collapse](#)

# On Footer Click

Code	Puede ser llamado por	Definición
57	<a href="#">List Box - Columna de List Box</a>	Se produce un clic en el pie de una columna de list box

## Descripción

Este evento está disponible para un objeto list box o columna de list box. Se genera cuando se produce un clic en el pie de una columna del list box. En este contexto, el comando `OBJECT Get pointer` devuelve un puntero a la variable del pie de página que se ha presionado. El evento se genera tanto para los clics izquierdos como para los derechos.

Puede probar el número de clics realizados por el usuario mediante el comando `Clickcount`.

# On Getting focus

Code	Puede ser llamado por	Definición
15	Área 4D View Pro - Área 4D Write Pro - Botón - Casilla de selección - Combo Box - Formulario - Lista jerárquica - Área de entrada - List Box - Columna de List Box - Área de plug-in - Indicador de progreso - Botón radio - Regla - Spinner - Stepper - Subformulario - Área Web	Un objeto formulario recibe el foco

## Descripción

Los eventos `On Getting Focus` y `On losing Focus`, se utilizan para detectar y manejar el cambio de foco de los objetos **focalizables**.

Con los **objetos subformulario**, este evento se genera en el método del objeto subformulario cuando se verifica. Se envía al método formulario del subformulario, lo que significa, por ejemplo, que se puede gestionar la visualización de los botones de navegación en el subformulario en función del foco. Tenga en cuenta que los objetos de subformulario pueden tener ellos mismos el foco.

# On Header

Code	Puede ser llamado por	Definición
5	Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Lista desplegable - Formulario (formulario lista únicamente) - Lista jerárquica - Área de entrada - Botón imagen - Pop up menu image - Área de plug-in - Indicador de progreso - Botón radio - Regla - Spinner - Splitter - Stepper - Pestaña	El área del encabezado del formulario está a punto de imprimirse o mostrarse.

## Descripción

El evento `On Header` se llama cuando un registro está a punto de ser visualizado en un formulario de lista que se muestra vía `DISPLAY SELECTION` y `MODIFY SELECTION`.

Este evento no se puede seleccionar para los formularios proyecto, sólo está disponible con los formularios tabla.

En este contexto, se desencadena la siguiente secuencia de llamadas a métodos y eventos de formulario:

- Para cada objeto en el área del encabezado:
  - Método objeto con el evento `On Header`
  - Método formulario con el evento `On Header`

Los registros impresos se gestionan mediante el evento `On Display Detail`.

Llamar a un comando 4D que muestra una caja de diálogo desde el evento `On Header` no está permitido y provocará un error de sintaxis. Más concretamente, los comandos en cuestión son: `ALERT`, `DIALOG`, `CONFIRM`, `Request`, `ADD RECORD`, `MODIFY RECORD`, `DISPLAY SELECTION` y `MODIFY SELECTION`.

# On Header Click

Code	Puede ser llamado por	Definición
42	<a href="#">Área 4D View Pro</a> - <a href="#">List Box</a> - <a href="#">Columna de List Box</a>	Se produce un clic en el encabezado de columna

## Descripción

### List Box

Este evento se genera cuando se hace clic en el encabezado de una columna de list box. En este caso, el comando `Self` permite identificar el encabezado de la columna sobre la que se ha hecho clic.

Si se seleccionó la propiedad [Sortable](#) para el list box, se puede decidir si se autoriza o no una ordenación estándar de la columna pasando el valor 0 o -1 en la variable `$0` :

- Si `$0` es igual a 0, se realiza una ordenación estándar.
- Si `$0` es igual a -1, no se realiza una ordenación estándar y el encabezado no muestra la flecha de ordenación. El desarrollador puede seguir generando una ordenación de columnas basada en criterios de ordenación personalizados utilizando el lenguaje 4D.

Si la propiedad [Sortable](#) no está seleccionada para el list box, la variable `$0` no se utiliza.

### 4D View Pro

Este evento se genera cuando el usuario hace clic en el encabezado de una columna o línea en un documento 4D View Pro. En este contexto, el [objeto evento](#) devuelto por el comando `FORM Event` contiene:

Propiedad	Tipo	Descripción
code	entero largo	42
description	texto	"On Header Click"
objectName	texto	Nombre del área 4D View Pro
sheetName	texto	Nombre de la hoja del evento
range	objeto	Rango de celdas
sheetArea	entero largo	La ubicación de la hoja donde tuvo lugar el evento: <ul style="list-style-type: none"><li>• 0: el área de cruce entre el número de columna/encabezado de letra (parte superior izquierda de la hoja)</li><li>• 1: los encabezados de las columnas (área que indica los números/letras de las columnas)</li><li>• 2: los encabezados de línea (área que indica los números de línea)</li></ul>

## Ejemplo

```
If(FORM Event.code=On Header Click)
Case of
:(FORM Event.sheetArea=1)
    $values:=VP Get values(FORM Event.range)
:(FORM Event.sheetArea=2)
    VP SET CELL STYLE(FORM Event.range;New object("backColor";"gray"))
:(FORM Event.sheetArea=0)
    VP SET CELL STYLE(FORM Event.range;New object("borderBottom";\
        New object("color";"#800080";"style";vk line style thick)))
End case
End if
```

# On Load

Code	Puede ser llamado por	Definición
1	Área 4D View Pro - Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Combo Box - Lista desplegable - Formulario - Lista jerárquica - Entrada - List Box - Columna de List Box - Botón imagen - Pop up menu imagen - Área de plug-in - Indicador de progreso - Botón radio - Regla - Spinner - Splitter - Stepper - Subformulario - Pestaña - Área Web	El formulario está a punto de ser mostrado o impreso

## Descripción

Este evento se activa cuando el formulario se está cargando o imprimiendo.

Todos los objetos del formulario (de cualquier página) cuya propiedad de evento `On Load` esté seleccionada tendrán su método objeto llamado. Entonces, si se selecciona la propiedad de evento formulario `On Load`, se llamará al método formulario.

Los eventos `On Load` y `On Unload` se generan para los objetos si están activados tanto para los objetos como para el formulario al que pertenecen los objetos. Si los eventos están activados sólo para los objetos, no se producirán; estos dos eventos también deben estar activados a nivel del formulario.

## Subformulario

El evento `On Load` se genera al abrir el subformulario (este evento debe haberse activado también a nivel del formulario padre para que se tenga en cuenta). El evento se genera antes que los del formulario padre. Tenga en cuenta también que, de acuerdo con los principios de funcionamiento de los eventos de formulario, si el subformulario se coloca en una página distinta de la página 0 o 1, este evento sólo se generará cuando se muestre esa página (y no cuando se muestre el formulario).

## Ver también

[On Unload](#)

# On Load Record

Code	Puede ser llamado por	Definición
40	Formulario	Durante la entrada del usuario en la lista, se carga un registro y se edita un campo

## Descripción

El evento `On Load Record` sólo puede utilizarse en el contexto de un formulario de salida. Se activa durante la entrada de datos en la lista, después de que se resalte un registro y un campo pase al modo de edición.

Este evento no se puede seleccionar para los formularios proyecto, sólo está disponible con los formularios tabla.

# On Long Click

Code	Puede ser llamado por	Definición
39	<a href="#">Botón</a>	Se presiona un botón y el botón del ratón permanece presionado durante un tiempo determinado

## Descripción

Este evento se genera cuando un botón recibe un clic y el botón del ratón se mantiene presionado durante un cierto tiempo. En teoría, el tiempo durante el cual se genera este evento es igual al tiempo máximo que separa un doble clic, definido en las preferencias del sistema.

Este evento se puede generar para los siguientes estilos de botones:

- [Barra de herramientas](#)
- [Bevel](#)
- [Bevel redondeado](#)
- [OS X Gradient](#)
- [OS X Texturizado](#)
- [Office XP](#)
- [Ayuda](#)
- [Círculo](#)
- [Personalizado](#)

Este evento se utiliza generalmente para mostrar menús emergentes en caso de presiones prolongadas en los botones. El evento [On Clicked](#), si está activo, se genera si el usuario suelta el botón del ratón antes del límite de tiempo del "clic largo".

## Ver también

[On Alternative Click](#)

# On Losing focus

Code	Puede ser llamado por	Definición
14	Área 4D View Pro - Área 4D Write Pro - Botón - Casilla de selección - Combo Box - Formulario - Lista jerárquica - Área de entrada - List Box - Columna de List Box - Área de plug-in - Indicador de progreso - Botón radio - Regla - Spinner - Stepper - Subformulario - Área Web	Un objeto formulario pierde el foco

## Descripción

Los eventos `On Losing Focus` y `On Getting Focus`, se utilizan para detectar y manejar el cambio de foco de los objetos **focalizables**.

Con los **objetos subformulario**, este evento se genera en el método del objeto subformulario cuando se verifica. Se envía al método formulario del subformulario, lo que significa, por ejemplo, que se puede gestionar la visualización de los botones de navegación en el subformulario en función del foco. Tenga en cuenta que los objetos de subformulario pueden tener ellos mismos el foco.

# On Menu Selected

Code	Puede ser llamado por	Definición
18	Formulario	Se ha elegido un elemento de menú en la barra de menús asociada

## Descripción

El evento `On Menu Selected` se envía al método del formulario cuando se selecciona un comando de una barra de menú asociada al formulario. A continuación, puede llamar al comando `Menú seleccionado` para probar el menú seleccionado.

Puede asociar una barra de menú a un formulario en las propiedades del mismo. Los menús de una barra de menús de un formulario se añaden a la barra de menús actual cuando el formulario se muestra como un formulario de salida en el entorno Aplicación.

# On Mouse Enter

Code	Puede ser llamado por	Definición
35	Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Combo Box - Lista desplegable - Formulario - Lista jerárquica - Entrada - List Box - Botón imagen - Pop up menu imagen - Área de plug-in - Indicador de progreso - Botón radio - Regla - Spinner - Splitter - Stepper - Pestaña	El cursor del ratón entra en el área gráfica de un objeto

## Descripción

Este evento se genera una vez, cuando el cursor del ratón entra en el área gráfica de un objeto del formulario.

El evento `On Mouse Enter` actualiza las variables sistema `MouseX` y `MouseY`.

Los objetos que se hacen invisibles utilizando el comando `OBJECT SET VISIBLE` o la propiedad `Visibilidad` no generan este evento.

## Llamar la pila

Si se ha marcado el evento `On Mouse Enter` para el formulario, se genera para cada objeto de formulario. Si se verifica para un objeto, se genera sólo para ese objeto. Cuando hay objetos superpuestos, el evento es generado por el primer objeto capaz de gestionarlo que se encuentra yendo en orden del nivel superior al inferior.

## Ver también

- [On Mouse Move](#)
- [On Mouse Leave](#)

# On Mouse Leave

Code	Puede ser llamado por	Definición
36	Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Combo Box - Lista desplegable - Formulario - Lista jerárquica - Entrada - List Box - Botón imagen - Pop up menu imagen - Área de plug-in - Indicador de progreso - Botón radio - Regla - Spinner - Splitter - Stepper - Pestaña	El cursor del ratón sale del área gráfica de un objeto

## Descripción

Este evento se genera una vez cuando el cursor del ratón abandona el área gráfica de un objeto.

El evento `On Mouse Leave` actualiza las variables sistema `MouseX` y `MouseY`.

Los objetos que se hacen invisibles utilizando el comando `OBJECT SET VISIBLE` o la propiedad `Visibilidad` no generan este evento.

## Llamar la pila

Si se ha marcado el evento `On Mouse Leave` para el formulario, se genera para cada objeto de formulario. Si se verifica para un objeto, se genera sólo para ese objeto. Cuando hay objetos superpuestos, el evento es generado por el primer objeto capaz de gestionarlo que se encuentra yendo en orden del nivel superior al inferior.

## Ver también

- [On Mouse Move](#)
- [On Mouse Leave](#)

# On Mouse Move

Code	Puede ser llamado por	Definición
37	Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Combo Box - Lista desplegable - Formulario - Lista jerárquica - Entrada - List Box - Botón imagen - Pop up menu imagen - Área de plug-in - Indicador de progreso - Botón radio - Regla - Spinner - Splitter - Stepper - Pestaña	El cursor del ratón se mueve al menos un píxel O se ha presionado una tecla de modificación (Shift, Alt/Option, Shift Lock)

## Descripción

Se genera este evento:

- cuando el cursor del ratón se mueve al menos un píxel
- O cuando se ha presionado una tecla de modificación ( Mayús, Alt/Opción, Bloq Mayús). Esto permite gestionar las operaciones de arrastrar y soltar de tipo copiar o mover.

Si el evento se marca para un objeto solamente, se genera sólo cuando el cursor está dentro del área gráfica del objeto.

El evento `On Mouse Move` actualiza las variables sistema `MouseX` y `MouseY`.

Los objetos que se hacen invisibles utilizando el comando `OBJECT SET VISIBLE` o la propiedad `Visibilidad` no generan este evento.

## Llamar la pila

Si se ha marcado el evento `On Mouse Move` para el formulario, se genera para cada objeto de formulario. Si se verifica para un objeto, se genera sólo para ese objeto. Cuando hay objetos superpuestos, el evento es generado por el primer objeto capaz de gestionarlo que se encuentra yendo en orden del nivel superior al inferior.

## Ver también

- [On Mouse Enter](#)
- [On Mouse Leave](#)

# On Mouse Up

Code	Puede ser llamado por	Definición
2	<a href="#">Área de entrada</a> de <a href="#">Tipo imagen</a>	El usuario acaba de soltar el botón izquierdo del ratón en un objeto Imagen

## Descripción

El evento `On Mouse Up` se genera cuando el usuario acaba de soltar el botón izquierdo del ratón mientras arrastra una imagen. Este evento es útil, por ejemplo, cuando se desea que el usuario pueda mover, redimensionar o dibujar objetos en un área SVG.

Cuando se genera el evento `On Mouse Up`, puede obtener las coordenadas locales donde se soltó el botón del ratón. Estas coordenadas se devuelven en las variables sistema `MouseX` y `MouseY`. Las coordenadas se expresan en píxeles con respecto a la esquina superior izquierda de la imagen (0,0).

Cuando se utiliza este evento, también hay que utilizar el comando `Is waiting mouse up` para manejar los casos en los que el "gestor de estado" del formulario está desincronizado, es decir, cuando el evento `On Mouse Up` no se recibe después de un clic. Este es el caso, por ejemplo, cuando se muestra una caja de diálogo de alerta sobre el formulario mientras no se ha soltado el botón del ratón. Para más información y un ejemplo de uso del evento `On Mouse Up`, consulte la descripción del comando `Is waiting mouse up`.

Si la opción [Draggable](#) está activada para el objeto `Imagen`, el evento `On Mouse Up` nunca se genera.

# On Open Detail

Code	Puede ser llamado por	Definición
25	Formulario - <a href="#">List Box</a>	El formulario detallado asociado al formulario de salida o al list box está a punto de abrirse.

## Descripción

El evento `On Open Detail` puede utilizarse en los siguientes contextos:

- Formularios de salida: un registro está a punto de ser mostrado en el formulario detallado asociado al formulario de salida. Este evento no se puede seleccionar para los formularios proyecto, sólo está disponible con los formularios tabla.
- List box [de tipo selección](#): este evento se genera cuando se va a mostrar un registro en el formulario detallado asociado a un list box del tipo de selección (y antes de que se abra este formulario).

## Número de línea mostrado

El comando 4D `Número de línea mostrado` funciona con el evento formulario `On Open Detail`. Devuelve el número de la línea que se está procesando mientras se visualiza en pantalla una lista de registros o de líneas de list box.

# On Open External Link

Code	Puede ser llamado por	Definición
52	<a href="#">Área Web</a>	Se ha abierto una URL externa en el navegador

## Descripción

Este evento se genera cuando la carga de una URL fue bloqueada por el área web y la URL fue abierta con el navegador actual del sistema, debido a un filtro configurado a través del comando `WA SET EXTERNAL LINKS FILTERS`.

Puede identificar la URL bloqueada utilizando el comando `WA Get last filtered URL`.

## Ver también

[On URL Filtering](#)

# On Outside Call

Code	Puede ser llamado por	Definición
10	Formulario	El formulario ha recibido una llamada <code>POST OUTSIDE CALL</code>

## Descripción

Este evento se llama cuando el formulario es llamado desde otro proceso a través del comando `POST OUTSIDE CALL`.

El evento `On Outside Call` modifica el contexto de entrada del formulario. En concreto, si se estaba editando un campo, se genera el evento `On Data Change`.

# On Page Change

Code	Puede ser llamado por	Definición
56	Formulario	Se cambia la página actual del formulario

## Descripción

Este evento sólo está disponible a nivel del formulario (se llama en el método formulario). Se genera cada vez que la página actual del formulario cambia (tras una llamada al comando `FORM GOT0 PAGE` o una acción de navegación estándar).

Note que se genera después de que la página esté completamente cargada, es decir, una vez que todos los objetos que contiene están inicializados, incluyendo [áreas web](#).

La única excepción son las áreas 4D View Pro, para las que hay que llamar al evento específico [On VP Ready](#).

El evento `On Page Change` es útil para ejecutar código que requiere que todos los objetos sean previamente inicializados. También se puede utilizar para optimizar la aplicación ejecutando el código (por ejemplo, una búsqueda) sólo después de que se muestre una página específica del formulario y no tan pronto como se cargue la página 1. Si el usuario no va a esta página, el código no se ejecuta.

# On Plug in Area

Code	Puede ser llamado por	Definición
19	Formulario - <a href="#">Área de plug-in</a>	Un objeto externo solicitó la ejecución de su método objeto

## Descripción

El evento se genera cuando un plug-in solicita su área de formulario para ejecutar el método objeto asociado.

# On Printing Break

Code	Puede ser llamado por	Definición
6	Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Combo Box - Lista desplegable - Formulario Lista jerárquica - Área de entrada - Botón imagen - Pop up menu image - Área de plug-in - Indicador de progreso - Botón radio - Regla - Spinner - Splitter - Stepper - Pestaña	Una de las áreas de ruptura del formulario está a punto de imprimirse

## Descripción

El evento `On Printing Break` sólo puede utilizarse en el contexto de un formulario de salida. Se activa cada vez que un área de ruptura del formulario de salida está a punto de imprimirse, para poder evaluar los valores de ruptura, por ejemplo.

Este evento suele producirse tras una llamada al comando `Subtotal`.

Este evento no se puede seleccionar para los formularios proyecto, sólo está disponible con los formularios tabla.

# On Printing Detail

Code	Puede ser llamado por	Definición
23	Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Combo Box - Lista desplegable - Formulario Lista jerárquica - Área de entrada - Botón imagen - Pop up menu image - Área de plug-in - Indicador de progreso - Botón radio - Regla - Spinner - Splitter - Stepper - Pestaña	El área detallada del formulario está a punto de imprimirse

## Descripción

El evento `On Printing Detail` sólo puede utilizarse en el contexto de un formulario de salida. Se activa cuando el área de detalle del formulario de salida está a punto de imprimirse, por ejemplo tras una llamada al comando `Print form`.

El comando `Print form` genera sólo un evento `On Printing Detail` para el método formulario.

Este evento no se puede seleccionar para los formularios proyecto, sólo está disponible con los formularios tabla.

# On Printing Footer

Code	Puede ser llamado por	Definición
7	Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Combo Box - Lista desplegable - Formulario Lista jerárquica - Área de entrada - Botón imagen - Pop up menu image - Área de plug-in - Indicador de progreso - Botón radio - Regla - Spinner - Splitter - Stepper - Pestaña	El área de pie de página del formulario está a punto de imprimirse

## Descripción

El evento `On Printing Footer` sólo puede utilizarse en el contexto de un formulario de salida. Se activa cuando el área de pie de página del formulario de salida está a punto de imprimirse, para que pueda evaluar los valores del pie de página.

Este evento puede activarse en el contexto de un comando `PRINT SELECTION`.

Este evento no se puede seleccionar para los formularios proyecto, sólo está disponible con los formularios tabla.

# On Resize

Code	Puede ser llamado por	Definición
29	Formulario	La ventana del formulario se redimensiona o el objeto subformulario se redimensiona (en este caso el evento se genera en el método formulario del subformulario)

## Descripción

Este evento se llama:

- cuando se redimensiona la ventana del formulario,
- en el contexto de los subformularios, cuando el tamaño del objeto subformulario en el formulario padre ha cambiado. En este caso, este evento se envía al método formulario del subformulario.

# On Row Moved

Code	Puede ser llamado por	Definición
34	<a href="#">List Box de tipo array - Columna de List Box</a>	Una línea de list box es movida por el usuario por medio de arrastrar y soltar

## Descripción

Este evento se genera cuando una línea de list box ( [sólo tipo array](#)) es movida por el usuario mediante la función de arrastrar y soltar ([si se permite](#). No se genera si la línea se arrastra y luego se suelta en su ubicación inicial.

El comando `LISTBOX MOVED ROW NUMBER` devuelve la nueva posición de la línea.

# On Row Resize

Code	Puede ser llamado por	Definición
60	Área 4D View Pro	La altura de una línea es modificada por un usuario con el ratón

## Descripción

Este evento se genera cuando la altura de una línea es modificada por un usuario en un documento 4D View Pro. En este contexto, el [objeto evento](#) devuelto por el comando `F0RM Event` contiene:

Propiedad	Tipo	Descripción
code	entero largo	60
description	texto	"On Row Resize"
objectName	texto	Nombre del área 4D View Pro
sheetName	texto	Nombre de la hoja del evento
range	objeto	Rango de celdas de las líneas cuyo alto ha cambiado
header	booleano	True si la línea de la columna de encabezado (primera línea) se redimensiona, si no false

## Ejemplo

```
If(FORM Event.code=On Row Resize)
    VP SET CELL STYLE(FORM Event.range;New object("vAlign";vk vertical align top))
End if
```

# On Scroll

Code	Puede ser llamado por	Definición
59	<a href="#">Área de entrada</a> de tipo <a href="#">Imagen</a> - <a href="#">List Box</a>	El usuario se desplaza por el contenido de un objeto imagen o de un list box utilizando el ratón o el teclado.

## Descripción

Este evento puede generarse en el contexto de una entrada de imagen o de un list box.

Este evento se desencadena después de cualquier otro evento usuario relacionado con la acción de desplazamiento ([On Clicked](#), [On After Keystroke](#), etc.). El evento sólo se genera en el método objeto (no en el método formulario).

El evento se dispara cuando el desplazamiento es el resultado de una acción del usuario: utilizando las barras de desplazamiento y/o los cursores, utilizando la rueda del ratón o [el teclado](#). No se genera cuando el objeto se desplaza debido a la ejecución del comando `OBJECT SET SCROLL POSITION`.

### Entrada de imagen

El evento se genera en cuanto un usuario se desplaza por una imagen dentro de la entrada de imagen (campo o variable) que la contiene. Puede desplazar el contenido de un área de imagen cuando el tamaño del área es menor que su contenido y el [formato de visualización](#) es "Truncado (no centrado)".

### List box

El evento se genera en cuanto un usuario se desplaza por las líneas o columnas del list box.

# On Selection Change

Code	Puede ser llamado por	Definición
31	<a href="#">Área 4D View Pro</a> - <a href="#">Área 4D Write Pro</a> - <a href="#">Formulario</a> - <a href="#">Lista jerárquica</a> - <a href="#">Área de entrada</a> - <a href="#">List Box</a>	La selección en el objeto se modifica

## Descripción

Este evento puede generarse en diferentes contextos.

### 4D View Pro

Se modifica la selección actual de líneas o columnas. En este contexto, el [objeto evento](#) devuelto por el comando `FORM Event` contiene:

Propiedad	Tipo	Descripción
code	entero largo	31
description	texto	"On Selection Change"
objectName	texto	Nombre del área 4D View Pro
sheetName	texto	Nombre de la hoja del evento
oldSelections	objeto	Rango de celdas antes del cambio
newSelections	objeto	Rango de celdas luego del cambio

### Ejemplo

```
If(FORM Event.code=On Selection Change)
    VP SET CELL STYLE(FORM Event.oldSelections;New object("backColor";Null))
    VP SET CELL STYLE(FORM Event.newSelections;New object("backColor";"red"))
End if
```

### Formulario listado

El registro actual o la selección actual de líneas se modifica en un formulario listado.

### Lista jerárquica

Este evento se genera cada vez que se modifica la selección en la lista jerárquica tras un clic del ratón o una presión del teclado.

### Área de entrada y 4D Write Pro

La selección de texto o la posición del cursor en el área se modifica tras un clic o una presión de tecla.

### List box

Este evento se genera cada vez que se modifica la selección actual de líneas o de columnas del list box.

# On Timer

Code	Puede ser llamado por	Definición
27	Formulario	El número de graduaciones definido por el comando <code>SET TIMER</code> ha pasado

## Descripción

Este evento se genera sólo si el método formulario contiene una llamada previa al comando `SET TIMER`.

Cuando se selecciona la propiedad de evento formulario `On Timer`, sólo el método formulario recibirá el evento, no se llamará a ningún método objeto.

# On Unload

Code	Puede ser llamado por	Definición
24	Área 4D View Pro - Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Combo Box - Lista desplegable - Formulario - Lista jerárquica - Entrada - List Box - Columna de List Box - Botón imagen - Pop up menu imagen - Área de plug-in - Indicador de progreso - Botón radio - Regla - Spinner - Splitter - Stepper - Subformulario - Pestaña - Área Web	El formulario está a punto de salir y liberarse

## Descripción

Este evento se activa cuando el formulario es generado.

Todos los objetos del formulario (de cualquier página) cuya propiedad de evento `On Unload` esté seleccionada tendrán su método objeto llamado. Entonces, si se selecciona la propiedad de evento formulario `On Unload`, se llamará al método formulario.

Los eventos `On Load` y [ `On Unload` ] se generan para los objetos si están activados tanto para los objetos como para el formulario al que pertenecen los objetos. Si los eventos están activados sólo para los objetos, no se producirán; estos dos eventos también deben estar activados a nivel del formulario.

## Subformulario

El evento `On Unload` se genera al cerrar el subformulario (este evento debe haberse activado también a nivel del formulario padre para que se tenga en cuenta). El evento se genera antes que los del formulario padre. El evento se genera antes que los del formulario padre.

## Ver también

[On Load](#)

# On URL Filtering

Code	Puede ser llamado por	Definición
51	<a href="#">Área Web</a>	Una URL fue bloqueada por el área web

## Descripción

Este evento se genera cuando la carga de una URL es bloqueada por el área web debido a un filtro configurado mediante el comando `WA SET URL FILTERS`.

Puede identificar la URL bloqueada utilizando el comando `WA Get last filtered URL`.

## Ver también

[On Open External Link](#)

# On URL Loading Error

Code	Puede ser llamado por	Definición
50	<a href="#">Área Web</a>	Se ha producido un error al cargar la URL

## Descripción

Este evento se genera cuando se detecta un error durante la carga de una URL.

Puede llamar al comando `WA GET LAST URL ERROR` para obtener información sobre el error.

## Ver también

[On Open External Link](#)

# On URL Resource Loading

Code	Puede ser llamado por	Definición
48	<a href="#">Área Web</a>	Se carga un nuevo recurso en el área web

## Descripción

Este evento se genera cada vez que se carga un nuevo recurso (imagen, marco, etc.) en la página web actual.

La variable [Progresión](#) asociada al área permite conocer el estado actual de la carga.

## Ver también

[On Open External Link](#)

# On Validate

Code	Puede ser llamado por	Definición
3	Área 4D Write Pro - Botón - Rejilla de botones - Casilla de selección - Combo Box - Lista desplegable - Formulario - Lista jerárquica - Entrada - List Box - Columna de List Box - Botón imagen - Pop up menu imagen - Área de plug-in - Indicador de progreso - Botón radio - Regla - Spinner - Splitter - Stepper - Subformulario - Pestaña	Se ha validado la entrada de datos del registro

## Descripción

Este evento se dispara cuando la entrada de datos del registro ha sido validada, por ejemplo después de una llamada al comando `SAVE RECORD` o una acción estándar `accept`.

## Subformulario

El evento `On Validate` se activa cuando se valida la entrada de datos en el subformulario.

# On VP Range Changed

Code	Puede ser llamado por	Definición
61	Área 4D View Pro	El rango de celdas 4D View Pro ha cambiado (por ejemplo, un cálculo de fórmula, un valor eliminado de una celda, etc.)

## Descripción

Este evento se genera cuando se produce un cambio dentro de un rango de celdas en el documento 4D View Pro.

El objeto devuelto por el comando FORM Event contiene:

Propiedad	Tipo	Descripción
objectName	texto	Nombre del área 4D View Pro
code	entero largo	On VP Range Changed
description	texto	"On VP Range Changed"
sheetName	texto	Nombre de la hoja del evento
range	objeto	Rango de celdas del cambio
changedCells	objeto	Rango que contiene sólo las celdas modificadas. Puede ser una gama combinada.
action	texto	El tipo de operación que genera el evento: <ul style="list-style-type: none"><li>• "clear" - operación para borrar el valor de un rango</li><li>• "dragDrop" - Una operación de arrastrar y soltar</li><li>• "dragFill" - Una operación de llenado por arrastre</li><li>• "evaluateFormula" - Definición de una fórmula en un rango de celdas especificado</li><li>• "pegar" - Una operación de pegado</li><li>• "setArrayFormula" - Definición de una fórmula en un rango de celdas especificado</li><li>• "sort" - Ordenar un rango de celdas</li></ul>

Ver también [On After Edit](#).

# On VP Ready

Code	Puede ser llamado por	Definición
9	<a href="#">Área 4D View Pro</a>	La carga del área 4D View Pro está completa

## Descripción

Este evento se genera cuando se completa la carga del área 4D View Pro.

Es necesario utilizar este evento para escribir el código de inicialización del área. Todo código de inicialización de área 4D View Pro, para cargar o leer valores desde o en el área, debe ubicarse en el evento formulario `On VP Ready` del área. Este evento formulario se activa una vez que se ha completado la carga del área. Probar este evento le asegura que el código se ejecutará en un contexto válido. Se devuelve un error si se llama a un comando 4D View Pro antes de que se genere el evento formulario `On VP Ready`.

Las áreas 4D View Pro se cargan de forma asíncrona en los formularios 4D. Esto significa que el evento formulario estándar `On load` no puede utilizarse para el código de inicialización de 4D View Pro, ya que podría ejecutarse antes de que se complete la carga del área. `On VP Ready` siempre se genera después de `On load`.

# On Window Opening Denied

Code	Puede ser llamado por	Definición
53	<a href="#">Área Web</a>	Se ha bloqueado una ventana emergente

► Histórico

## Descripción

Este evento se genera cuando la apertura de una ventana emergente es bloqueada por el área web. Los áreas web de 4D no permiten la apertura de ventanas emergentes.

Puede identificar la URL bloqueada utilizando el comando [WA Get last filtered URL](#).

This event is also triggered when a drop operation has been done in the Web area (with embedded and Windows system engines) if the [Drag and drop](#) option is also enabled for the area. Puede aceptar la acción de soltar llamando:

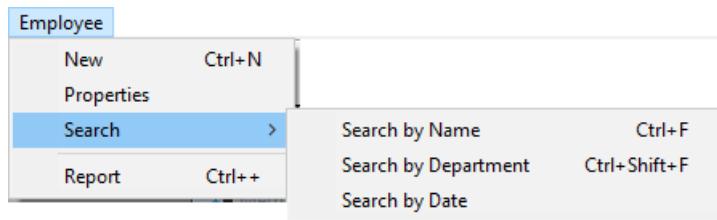
```
//web area object method
If (FORM Event.code=On Window Opening Denied)
  WA OPEN URL(*; "WebArea"; WA Get last filtered URL(*; "WebArea"))
  // or UrlVariable:=WA Get last filtered URL(*; "WebArea")
  // where UrlVariable is the URL variable associated to the web area
End if
```

## Ver también

[On Open External Link](#)

# Generalidades

Puede crear barras de menús y menús para sus aplicaciones 4D. Dado que los menús desplegables son una funcionalidad estándar de toda aplicación de escritorio, su adición facilitará el uso de sus aplicaciones y hará que los usuarios se sientan familiares.



Una barra de menús es un grupo de menús que pueden aparecer juntos en una misma pantalla. Cada menú de una barra de menús puede tener varios comandos de menú en ella, incluyendo algunos que llaman a submenús en cascada (o submenús jerárquicos). Cuando el usuario elige un comando de menú o submenú, llama a un método proyecto o a una acción estándar que realiza una operación.

Puede tener varias barras de menús separadas para cada aplicación. Por ejemplo, puede utilizar una barra de menús que contenga menús para las operaciones estándar de la base de datos y otra que se active sólo para los informes. Una barra de menús puede contener un menú con comandos de menú para introducir registros. La barra de menús que aparece con el formulario de entrada puede contener el mismo menú, pero los comandos del menú están desactivados porque el usuario no los necesita durante la entrada de datos.

Puede utilizar el mismo menú en varias barras de menús o en otros menús, o puede dejarlo suelto y gestionarlo sólo por programación (en este caso, se conoce como menú independiente).

Cuando diseña los menús, tenga en cuenta las dos reglas siguientes:

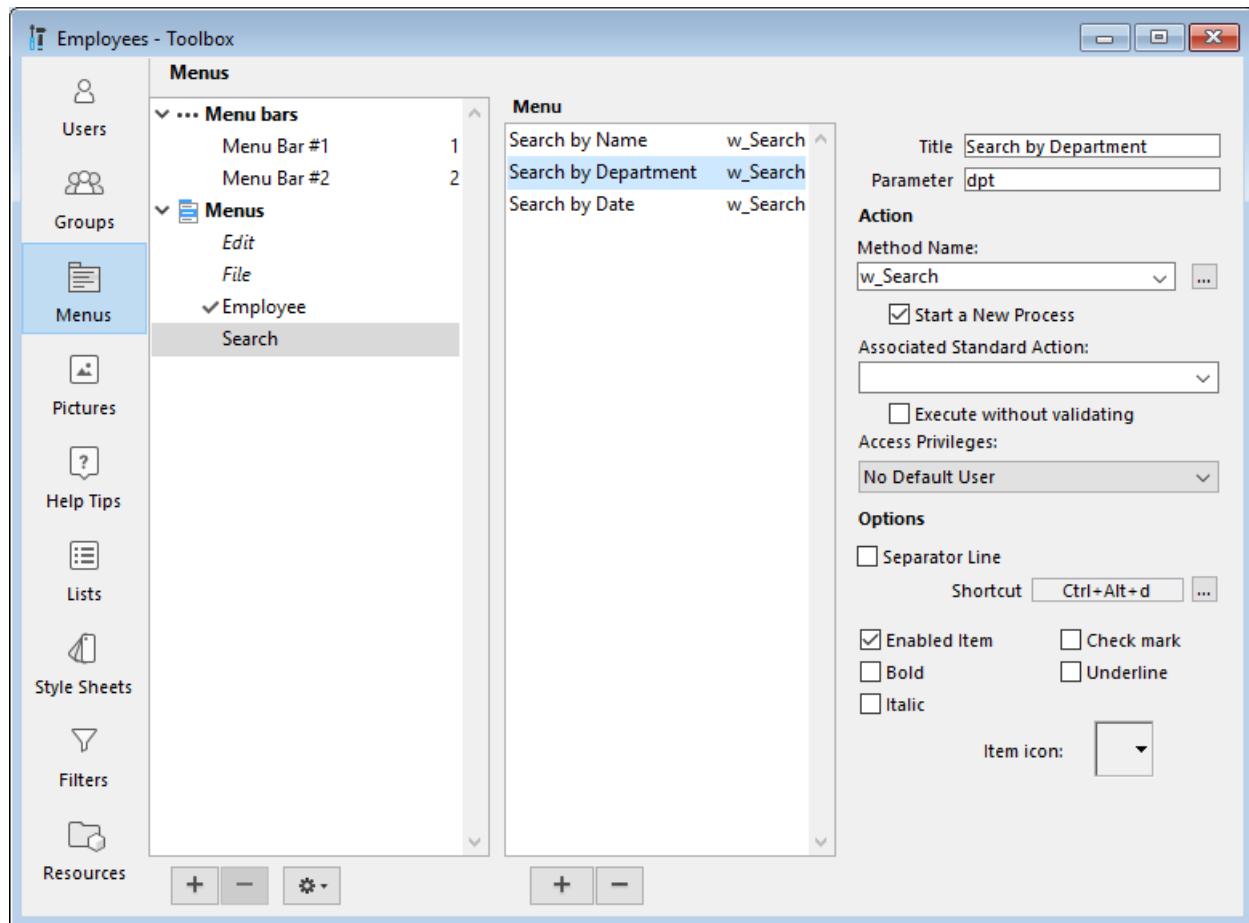
- Utilice los menús para las funciones que se adaptan a los menús: los comandos del menú deben realizar tareas como añadir un registro, buscar registros o imprimir un informe.
- Agrupe los comandos de menú por función: por ejemplo, todos los comandos de menú que imprimen informes deben estar en el mismo menú. Por ejemplo, puede tener todas las operaciones de una determinada tabla en un solo menú.

Para crear menús y barras de menús, puede utilizar:

- el editor de menús de la caja de herramientas,
- los comandos de lenguaje para el tema "Menús",
- una combinación de ambos.

## Editor de menús

Se accede al editor de menús mediante el botón Menús de la caja de herramientas.



Los menús y las barras de menús se muestran como dos elementos de la misma lista jerárquica, en la parte izquierda de la caja de diálogo. Cada menú puede estar unido a una barra de menús o a otro menú. En el segundo caso, el menú se convierte en un submenú.

4D asigna los números de las barras de menú de forma secuencial: Menu Bar #1 aparece primero. Puede cambiar el nombre de las barras de menú, pero no puede cambiar sus números. Estos números son utilizados por los comandos del lenguaje.

# Creación de menús y barras de menús

Puede crear menús y barras de menús:

- utilizando el editor de menús de la ventana de la caja de herramientas 4D. En este caso, los menús y las barras de menú se almacenan en la estructura de la aplicación.
- dinámicamente, utilizando los comandos del lenguaje del tema "Menús". En este caso, los menús y las barras de menús no se almacenan, sólo existen en la memoria.

Puede combinar ambas funcionalidades y utilizar los menús creados en la estructura como plantillas para definir los menús en la memoria.

## Barra de menús por defecto

Una aplicación personalizada debe contener al menos una barra de menús con un menú. Por defecto, cuando se crea un nuevo proyecto, 4D crea automáticamente una barra de menú por defecto (Barra de nº 1) para que pueda acceder al modo Aplicación. La barra de menús por defecto incluye menús estándar y un comando para volver al modo Diseño.

Esto permite al usuario acceder al modo Aplicación tan pronto como se crea el proyecto. La barra de menús nº 1 se llama automáticamente cuando se elige el comando Test Application en el menú Ejecución.

La barra de menús por defecto incluye tres menús:

- Archivo: sólo incluye el comando Salir. La acción estándar *Salir* está asociada al comando, que hace que la aplicación salga.
- Editar: menú estándar y totalmente editable. Las funciones de edición, como copiar, pegar, etc., se definen mediante acciones estándar.
- Modo: contiene, por defecto, el comando Volver al modo Diseño, que se utiliza para salir del modo Aplicación.

Los elementos del menú aparecen \*en itálica \* porque consisten de referencias y no de texto codificado. Consulte [Propiedad Título](#).

Puede modificar esta barra de menú como deseé o crear otras adicionales.

## Crear menús

### Utilizando el editor de menús

1. Seleccione el elemento que desea crear y haga clic en el botón añadir debajo de la zona de la barra de menús. O Elija Crear una nueva barra de menús o Crear un nuevo menú en el menú contextual de la lista o en el menú de opciones situado debajo de la lista. Si ha creado una barra de menús, aparece una nueva barra en la lista que contiene los menús por defecto (Archivo y Edición).
2. (opcional) Haga doble clic en el nombre de la barra de menú/menú para pasar al modo de edición e introducir un nombre personalizado. O Introduzca el nombre personalizado en el área "Título". Los nombres de las barras de menú deben ser únicos. Pueden contener hasta 31 caracteres. Puede introducir el nombre como "hard coded" o introducir una referencia (ver [información sobre la propiedad Title](#)).

### Utilizando el lenguaje 4D

Utilice el comando `Create menu` para crear una nueva barra de menú o referencia de menú (`MenuRef`) en la memoria.

Cuando los menús se manejan mediante referencias `MenuRef`, no hay diferencia per se entre un menú y una barra de menús. En ambos casos, consiste en una lista de elementos. Sólo difiere su uso. En el caso de una barra de menús,

cada elemento corresponde a un menú que a su vez está compuesto por elementos.

Create menu puede crear menús vacíos (para llenar utilizando APPEND MENU ITEM o INSERT MENU ITEM ) o por menús construidos sobre menús diseñados en el editor de menús.

## Añadir líneas

Para cada uno de los menús, debe añadir los comandos que aparecen cuando el menú se despliega. Puede insertar elementos que se asociarán a los métodos o a las acciones estándar, o adjuntar otros menús (submenús).

### Utilizando el editor de menús

Para añadir un elemento de menú:

1. En la lista de menús fuente, seleccione el menú al que desea añadir un comando. Si el menú ya tiene comandos, se mostrarán en la lista central. Si desea insertar el nuevo comando, seleccione el comando que desea que aparezca sobre él. Todavía es posible reorganizar el menú posteriormente utilizando la función de arrastrar y soltar.
2. Elija Añadir un elemento al menú "NombreDelMenú" en el menú de opciones del editor o desde el menú contextual (clic derecho en la lista central). O Haga clic en el botón añadir + situado debajo de la lista central. 4D añade un nuevo elemento con el nombre por defecto "Elemento X" donde X es el número de elementos ya creados.
3. Haga doble clic en el nombre del comando para pasar al modo edición e introducir un nombre personalizado. O Introduzca el nombre personalizado en el área "Título". Puede contener hasta 31 caracteres. Puede introducir el nombre como "codificado" o introducir una referencia (ver más abajo).

### Utilizando el lenguaje 4D

Utilice INSERT MENU ITEM o APPEND MENU ITEM para insertar o añadir elementos de menú en referencias de menú existentes.

## Eliminar menús y elementos de menús

### Utilizando el editor de menús

Puede eliminar una barra de menús, un menú o un elemento de menú en el editor de menús en cualquier momento. Tenga en cuenta que cada menú o barra de menús sólo tiene una referencia. Cuando un menú está unido a diferentes barras o a diferentes menús, cualquier modificación o supresión realizada en el menú se lleva a cabo inmediatamente en todas las demás apariciones de este menú. La eliminación de un menú sólo borrará una referencia. Cuando se borra la última referencia de un menú, 4D muestra una alerta.

Para eliminar una barra de menús, un menú o un elemento de menú:

- Seleccione el elemento que desea eliminar y haga clic en el botón - situado debajo de la lista.
- o bien, utilice el comando apropiado Eliminar... del menú contextual o del menú de opciones del editor.

No es posible eliminar Menu Bar #1.

### Utilizando el lenguaje 4D

Utilice DELETE MENU ITEM para eliminar un elemento de una referencia de menú. Utilice RELEASE MENU para descargar la referencia del menú de la memoria.

## Adjuntar los menús

Una vez creado un menú, puede adjuntarlo a uno o a varios otros menús (submenús) o barras de menús.

Los submenús pueden utilizarse para agrupar funciones organizadas por temas dentro del mismo menú. Los submenús y sus elementos pueden tener los mismos atributos que los propios menús (acciones, métodos, accesos directos, iconos, etc.). Los elementos del submenú conservan sus características y propiedades originales y el funcionamiento del

submenú es idéntico al de un menú estándar.

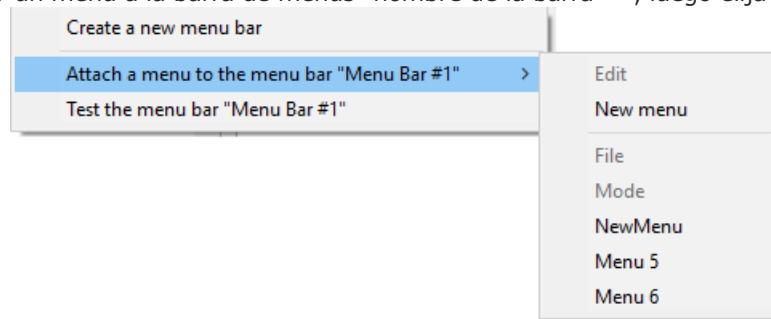
Puede crear submenús de submenús hasta una profundidad prácticamente ilimitada. Sin embargo, tenga en cuenta que, por razones de ergonomía de la interfaz, en general no se recomienda superar los dos niveles de submenús.

En tiempo de ejecución, si un menú adjunto se modifica por programación, todas las demás instancias del menú reflejarán estos cambios.

## Utilizando el editor de menús

Un menú puede estar unido a una barra de menús o a otro menú.

- Para adjuntar un menú a una barra de menús: haga clic con el botón derecho del ratón en la barra de menús y seleccione Adjuntar un menú a la barra de menús "nombre de la barra" >, luego elija el menú que desea adjuntar a

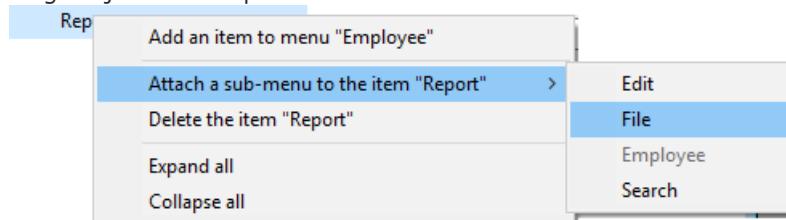


la barra de menús:

También puede seleccionar

una barra de menú y luego hacer clic en el botón de opciones que se encuentra debajo de la lista.

- Para adjuntar un menú a otro menú: seleccione el menú en el área de la izquierda, luego haga clic con el botón derecho del ratón en el elemento del menú y seleccione Agregar un submenú al elemento "nombre del elemento">>, luego elija el menú que desea utilizar como submenú:



También puede seleccionar un elemento del menú y luego hacer clic en el botón de opciones que se encuentra debajo de la lista. El menú que se adjunta se convierte así en un submenú. El título del elemento se mantiene (el nombre original del submenú se ignora), pero este título puede modificarse.

## Quitar los menús

Puede separar un menú de una barra de menús o un submenú de un menú en cualquier momento. El menú separado ya no está disponible en la barra de menús o en el submenú, según el caso, pero sigue estando presente en la lista de menús.

Para desvincular un menú, haga clic con el botón derecho en el menú o submenú que deseé desvincular en la lista central y, a continuación, elija la opción Desvincular el menú(...) o Desvincular el submenú(...)

## Utilizando el lenguaje 4D

Como no hay diferencia entre los menús y las barras de menús en el lenguaje de 4D, adjuntar menús o submenús se hace de la misma manera: utilice el parámetro *submenú* del comando APPEND MENU ITEM para adjuntar un menú a una barra de menús o a un menú.

# Propiedades de los menús

Puede definir varias propiedades para los elementos del menú, como las acción, el estilo de la fuente, las líneas de separación, los atajos de teclado o los iconos.

## Título

La propiedad Title contiene la etiqueta de un menú o elemento de menú tal y como se mostrará en la interfaz de la aplicación.

En el editor de menús, puede introducir directamente la etiqueta como "fijo". También puede introducir una referencia para una variable o un elemento XLIFF, lo que facilitará el mantenimiento y la traducción de las aplicaciones. Puede utilizar los siguientes tipos de referencias:

- Una referencia a un recurso XLIFF del tipo :xlfiff:MyLabel. Para más información sobre las referencias XLIFF, consulte la sección *Arquitectura XLIFF* en *Referencia Diseño 4D*.
- Un nombre de variable interproceso seguido de un número, por ejemplo: :<>vlang,3. Si se cambia el contenido de esta variable, se modificará la etiqueta del menú cuando se muestre. En este caso, la etiqueta llamará a un recurso XLIFF. El valor contenido en la variable <>vlang corresponde al atributo *id* del elemento *group*. El segundo valor (3 en este ejemplo) designa el atributo *id* del elemento *trans-unit*.

Utilizando el lenguaje 4D, se define la propiedad del título a través del parámetro *itemText* de los comandos APPEND MENU ITEM , INSERT MENU ITEM , y SET MENU ITEM .

## Caracteres de control

Puede definir algunas propiedades de los comandos de menú utilizando caracteres de control (metacaracteres) directamente en las etiquetas de los comandos de menú. Por ejemplo, puede asignar el atajo de teclado Ctrl+G (Windows) o Comando+G (macOS) para un comando de menú colocando los caracteres "/G" en la etiqueta del elemento de menú.

Los caracteres de control no aparecen en las etiquetas de los comandos del menú. Por lo tanto, hay que evitar su uso para no tener efectos indeseables. Los caracteres de control son los siguientes:

Carácter	Descripción	Utilización
(	paréntesis de apertura	Desactivar la línea
<B	menor que B	Negrita
<I	menor que I	Itálica
<U	menor que U	Subrayado
!+carácter	signo de exclamación+carácter	Carácter de adición como marca de verificación (macOS); añadir marca de verificación (Windows)
/+carácter	barra+carácter	Añadir un carácter como atajo

## Parámetros

Puede asociar un parámetro personalizado a cada elemento del menú. Un parámetro de elemento de menú es una cadena de caracteres cuyo contenido puede elegirse libremente. Puede definirse en el editor de menús, o a través del comando SET MENU ITEM PARAMETER .

Los parámetros de elementos de menú son útiles con la gestión programada de los menús, en particular cuando se utilizan los comandos Dynamic pop up menu , Get menu item parameter y Get selected menu item parameter .

## Acción

Cada comando de menú puede tener un método proyecto o una acción estándar adjunta. Cuando se elige el comando del menú, 4D ejecuta la acción estándar o el método proyecto asociado. Por ejemplo, un comando de menú Informe mensual puede llamar a un método proyecto que准备 un informe mensual a partir de una tabla que contenga datos financieros. El comando del menú Cortar suele llamar a la acción estándar `cut` para mover la selección al portapapeles y borrarla de la ventana en primer plano.

Si no asigna un método o una acción estándar a un comando de menú, la elección de ese comando de menú hace que 4D salga del entorno de Aplicación y pase al entorno de Diseño. Si sólo está disponible el entorno de la Aplicación, esto significa salir al Escritorio.

Las acciones estándar permiten realizar diversas operaciones cotidianas vinculadas a las funciones sistema (copiar, salir, etc.) o a las de la base (añadir registro, seleccionar todo, etc.).

Puede asignar al mismo tiempo una acción estándar y un método proyecto a un comando de menú. En este caso, la acción estándar no se ejecuta nunca; sin embargo, 4D utiliza esta acción para habilitar/deshabilitar el comando de menú según el contexto actual y para asociar una operación específica según la plataforma. Cuando se desactiva un comando de menú, no se puede ejecutar el método proyecto asociado.

La elección entre asociar una acción estándar o un método proyecto a un comando de menú depende del tipo de resultado deseado. En principio, es preferible elegir una acción estándar siempre que sea posible, ya que implementan mecanismos optimizados, más concretamente la activación/desactivación en función del contexto.

## Asociar un método proyecto o una acción estándar

Puede asignar un método proyecto y/o una acción estándar a un comando de menú seleccionado en el editor de menús:

- **Nombre del método:** seleccione un nombre de método proyecto existente en el combo box. Si el método proyecto no existe, introduzca su nombre en el combo "Nombre del método" y haga clic en el botón [...]. 4D muestra un diálogo de creación de método proyecto que se utiliza para acceder al editor de métodos.
- **Acción estándar asociada:** elija o escriba la acción que desea asignar en el combo box "Acción estándar asociada". Puede introducir toda acción soportada y (opcionalmente) el parámetro que desee en el área. Para obtener una lista completa de acciones estándar, consulte la sección Acciones estándar en el *Modo Diseño*. Nota para macOS: en macOS, los comandos de menús personalizados asociados a la acción *Salir* se colocan automáticamente en el menú de la aplicación, conforme a los estándares de interfaz de la plataforma.

Utilizando el lenguaje 4D, puedes asociar un método proyecto utilizando el comando `SET MENU ITEM METHOD`, y una acción estándar utilizando el comando `SET MENU ITEM PROPERTY`.

## Iniciar un proceso

La opción Iniciar un nuevo proceso está disponible para los comandos de menú asociados a los métodos. Puede definirse a través de una casilla de selección en el editor de menús, o a través del parámetro *propiedad* del comando `SET MENU ITEM PROPERTY`.

Cuando la opción Iniciar un nuevo proceso está activada, se crea un nuevo proceso cuando se elige el comando de menú. Normalmente, un método asociado a un comando de menú se ejecuta en el proceso actual a menos que llame explícitamente a un nuevo proceso en su código. La opción Iniciar un nuevo proceso facilita el lanzamiento de un nuevo proceso. Si se activa, 4D creará un nuevo proceso cuando se elija el comando del menú.

En la lista de procesos, 4D asigna al nuevo proceso un nombre por defecto utilizando el formato "ML\_NúmeroProceso". Los nombres de los procesos iniciados desde un menú se crean combinando el prefijo "ML\_" con el número del proceso.

## Ejecutar sin validar

La opción Ejecutar sin validar está disponible sólo para los comandos de menú asociados a acciones estándar en el editor de menús.

Cuando esta opción está marcada, 4D no activa la "validación" del campo donde se encuentra el cursor antes de ejecutar la acción asociada. Esta opción está pensada principalmente para los comandos del menú Edición. Por defecto, 4D procesa y "valida" el contenido de un campo antes de ejecutar una acción estándar (a través de un comando de menú o un acceso directo), que tiene el efecto de generar un evento de formulario `On Data Change`. Esto puede

interrumpir el funcionamiento de los comandos de tipo copiar o pegar, ya que cuando son llamados, el evento formulario `On Data Change` se genera inesperadamente. En este caso, es útil marcar la opción `Ejecutar sin validar`.

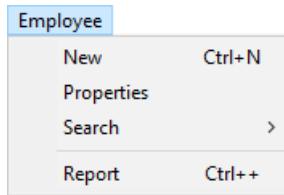
## Privilegios de acceso remoto

Esta opción del editor de menús permite definir un grupo para un comando de menú para que sólo los usuarios de ese grupo puedan utilizar el comando de menú desde una aplicación remota 4D (ver [Usuarios y grupos](#)).

## Opciones

### Líneas de separación

Los grupos de comandos de menús pueden estar divididos por una línea de separación. Esta convención es útil para agrupar los comandos de menú asociados por función.



Se añade una línea de separación mediante la creación de un comando de menú específico.

En el editor de menús, en lugar de introducir el texto del comando de menú en el área del título, basta con seleccionar la opción Línea separadora. En lugar de texto, aparece una línea en el área de la barra de menú actual. Cuando esta opción está marcada, las otras propiedades no tienen efecto. Nota: en macOS, si utiliza el guión "-" como primer carácter de un elemento de menú, aparecerá como línea de separación.

En el lenguaje 4D, se inserta una línea de separación introduciendo `-` o `(-` como `itemText` para los comandos `APPEND MENU ITEM`, `INSERT MENU ITEM`, o `SET MENU ITEM`.

### Atajos de teclado

Puede añadir atajos de teclado a todo comando de menú. Si un comando de menú tiene uno de estos atajos de teclado, los usuarios lo verán junto al comando de menú. Por ejemplo, "Ctrl+C" (Windows) o "Comando+C" (macOS) aparece junto al comando de menú Copiar en el menú Edición.

También puede añadir la tecla Mayús, así como la tecla Alt (Windows) u Opción (macOS) al acceso directo asociado a un comando de menú. Esto multiplica el número de accesos directos que se pueden utilizar. Por lo tanto, se pueden definir los siguientes tipos de atajos de teclado:

- En Windows:
  - Ctrl+caracter
  - Ctrl+Mayús+letra
  - Ctrl+Alt+caracter
  - Ctrl+Mayús+Alt+letra
- En macOS:
  - Comando+letra
  - Comando+Mayús+letra
  - Comando+Opción+letra
  - Comando+Mayús+Opción+letra

Le recomendamos que mantenga los atajos de teclado por defecto que están asociados a las acciones estándar.

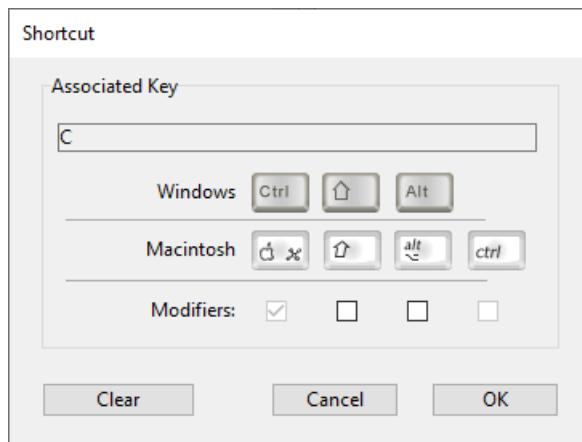
Puede utilizar cualquier tecla alfanumérica como atajo de teclado, excepto las teclas reservadas por los comandos de menú estándar que aparecen en los menús Edición y Archivo, y las teclas reservadas para los comandos de menú 4D.

Estas combinaciones de teclas reservadas se indican en la siguiente tabla:

Llave (Windows)	Llave (macOS)	Operación
Ctrl+C	Comando+C	Copiar
Ctrl+Q	Comando+Q	Salir
Ctrl+V	Comando+V	Pegar
Ctrl+X	Comando+X	Cortar
Ctrl+Z	Comando+Z	Deshacer
Ctrl+. (punto)	Comando+. (punto)	Detener la acción

Para asignar un atajo de teclado en el editor de menús:

Seleccione la opción de menú a la que desea asignar un atajo de teclado. Haga clic en el botón [...] situado a la derecha del área de entrada "Acceso directo". Aparece la siguiente ventana:



Introduzca el carácter que desea utilizar y, a continuación (opcional), haga clic en las casillas de selección Mayús y/o Alt (Opción) según la combinación deseada. También puede presionar directamente las teclas que componen la combinación deseada (no presione la tecla Ctrl/Comando).

No se puede desmarcar la tecla Ctrl/Comando, que es obligatoria para los atajos de teclado de los menús. Para volver a empezar, haga clic en Borrar. Haga clic en Aceptar para validar los cambios. El acceso directo definido se muestra en el área de entrada "Acceso directo".

Para asignar un atajo de teclado utilizando el lenguaje 4D, utilice el comando `SET ITEM SHORTCUT`.

Un objeto activo también puede tener un atajo de teclado. Si las asignaciones de las teclas Ctrl/Comando entran en conflicto, el objeto activo tiene prioridad.

## Línea activa

En el editor de menús, puede especificar si un elemento del menú aparecerá activo o inactivo. Un comando de menú activo puede ser elegido por el usuario; un comando de menú inactivo está atenuado y no puede ser elegido. Cuando la casilla de selección Línea activa está desmarcada, el comando de menú aparece atenuado, lo que indica que no se puede elegir.

A menos que especifique lo contrario, 4D habilita automáticamente cada elemento de menú que añada a un menú personalizado. Puede desactivar un elemento para, por ejemplo, activarlo sólo por programación con los comandos `ENABLE MENU ITEM` y `DISABLE MENU ITEM`.

## Marca de verificación

Esta opción del editor de menús puede utilizarse para asociar una marca de verificación del sistema a un elemento del menú. A continuación, puede gestionar la visualización de la marca de verificación utilizando los comandos del lenguaje

( SET MENU ITEM MARK y Get menu item mark ).

Las marcas de verificación se utilizan generalmente para los elementos del menú de acción continua e indican que la acción está en curso.

## Estilos de fuentes

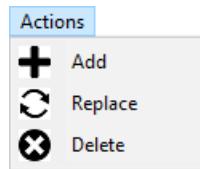
4D le permite personalizar los menús aplicando diferentes estilos de letra a los comandos del menú. Puede personalizar sus menús con los estilos Negrita, Cursiva o Subrayado a través de las opciones del editor de menús, o utilizando el comando de lenguaje SET MENU ITEM STYLE .

Como regla general, aplique los estilos de fuente con moderación a sus menús; demasiados estilos distraerán al usuario y darán un aspecto desordenado a su aplicación.

También puede aplicar estilos insertando caracteres especiales en el título del menú (ver [Usar caracteres de control](#) más arriba).

## Icono línea

Puede asociar un ícono a un elemento del menú. Aparecerá directamente en el menú, junto al elemento:



Para definir el ícono en el editor de menús, haga clic en el área "Icono del elemento" y seleccione Abrir para abrir una imagen del disco. Si selecciona un archivo imagen que no está almacenado en la carpeta Recursos del proyecto, se copiará automáticamente en esa carpeta. Una vez definido, el ícono del elemento aparece en el área de vista previa:



Para eliminar el ícono del elemento, elija la opción Sin ícono del área "Icono línea".

Para definir los íconos de los elementos utilizando el lenguaje 4D, llame al comando SET MENU ITEM ICON .

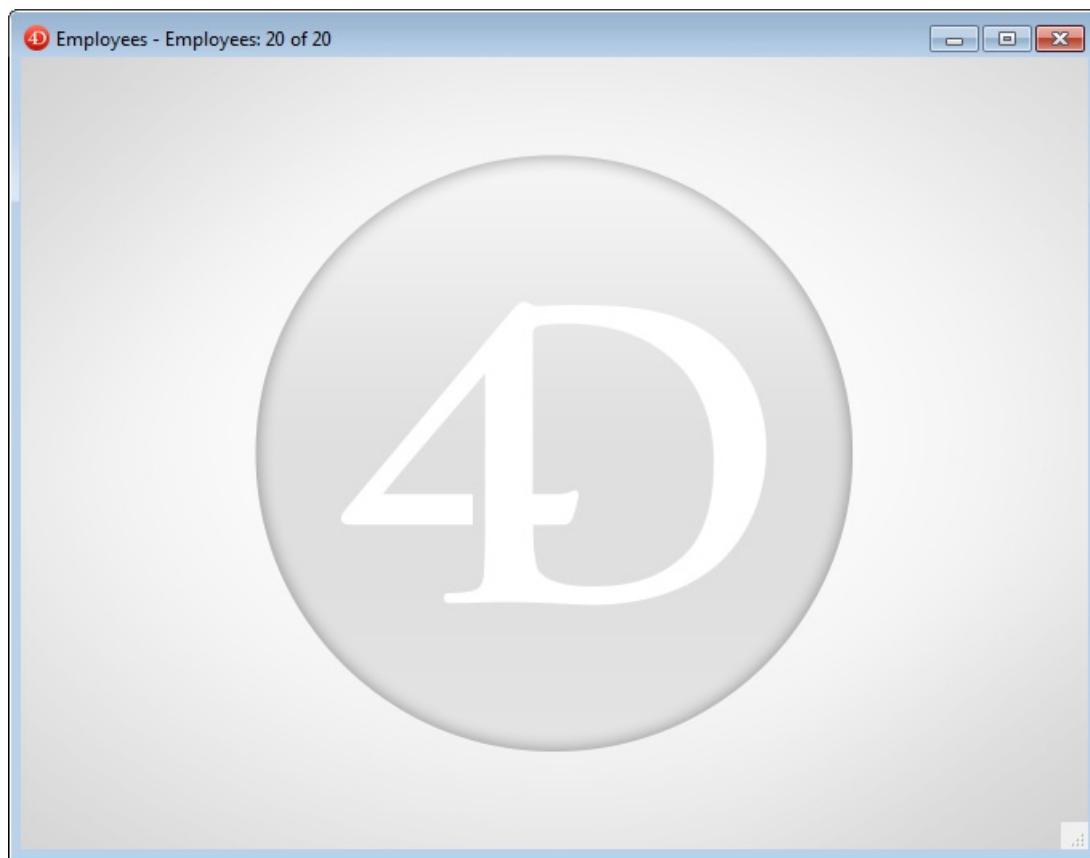
# Barras de menús

Las barras de menú constituyen la principal interfaz de las aplicaciones personalizadas. Para cada aplicación personalizada, debe crear al menos una barra de menú con al menos un menú. Por defecto, Menu Bar #1 es la barra de menús que se muestra en el entorno de la aplicación. Puede cambiar la barra de menús que se muestra utilizando el comando `SET MENU BAR`.

4D le permite asociar una imagen de pantalla de inicio personalizada con cada barra de menús y previsualizar esta barra de menú en cualquier momento.

## Pantalla de bienvenida

Puede mejorar la apariencia de cada barra de menú asociando una pantalla de inicio personalizada. La ventana que contiene la pantalla de inicio se muestra debajo de la barra de menús cuando aparece. Puede contener un logo o cualquier tipo de imagen. Por defecto, 4D muestra el logo 4D en la pantalla de inicio:



Una imagen de pantalla de inicio personalizada puede provenir de cualquier aplicación gráfica. 4D le permite pegar una imagen del portapapeles o utilizar cualquier imagen presente en su disco duro. Se puede utilizar cualquier formato de imagen estándar soportado por 4D.

La imagen de la pantalla de inicio sólo puede definirse en el editor de menús: seleccione la barra de menús a la que desea asociar la pantalla de inicio personalizada. Observe el área "Imagen de fondo" en la parte derecha de la ventana. Para abrir directamente una imagen almacenada en su disco, haga clic en el botón Abrir o en el área "Imagen de fondo". Aparece un menú emergente:

- Para pegar una imagen desde el portapapeles, seleccione Pegar.
- Para abrir una imagen almacenada en un archivo de disco, seleccione Abrir. Si selecciona Abrir, aparecerá una caja de diálogo estándar de Abrir archivo para que pueda seleccionar el archivo de imagen que va a utilizar. Una vez definida, la imagen se muestra en miniatura en la zona. A continuación, se asocia a la barra de menús.



Puede ver el resultado final probando la barra de menús (ver la sección siguiente). En el modo Aplicación, la imagen se muestra en la pantalla de inicio con el formato de tipo "Truncado (Centrado)".

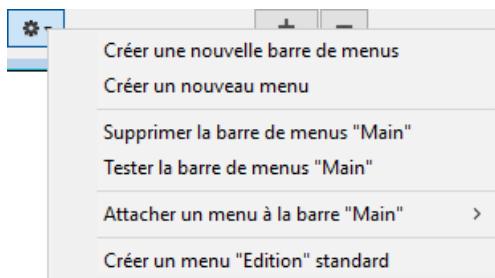
Puede elegir si desea mostrar u ocultar esta ventana mediante la opción Mostrar la barra de herramientas en las Propiedades.

Para eliminar la imagen personalizada y mostrar la predeterminada en su lugar, haga clic en el botón Borrar o seleccione Borrar en el menú emergente del área.

## Vista previa de las barras de menús

El editor de la barra de menús le permite ver los menús personalizados y la pantalla de inicio en cualquier momento, sin cerrar la ventana de la caja de herramientas.

Para ello, basta con seleccionar la barra de menús y elegir Probar la barra de menús "Barra de menús #X" en el menú contextual o en el menú de opciones del editor.



4D muestra una vista previa de la barra de menús así como de la pantalla de inicio. Puede desplazarse por los menús y submenús para ver su contenido. Sin embargo, estos menús no están activos. Para probar el funcionamiento de los menús y la barra de herramientas, debe utilizar el comando Probar la aplicación en el menú Ejecutar.

# Modo SDI bajo Windows

En Windows, los desarrolladores 4D pueden configurar sus aplicaciones fusionadas 4D para que funcionen como aplicaciones SDI (Single-Document Interface). En las aplicaciones SDI, cada ventana es independiente de las demás y puede tener su propia barra de menús. Las aplicaciones SDI se oponen a las aplicaciones MDI (Multiple Documents Interface), en las que todas las ventanas están contenidas y dependen de la ventana principal.

El concepto de SDI/MDI no existe en macOS. Esta funcionalidad sólo afecta a las aplicaciones de Windows y las opciones relacionadas se ignoran en macOS.

## Disponibilidad del modo SDI

El modo SDI sólo está disponible en el siguiente entorno de ejecución:

- Windows
- Aplicación 4D fusionada, monopuesto o cliente

## Activación del modo SDI

La activación y el uso del modo SDI en su aplicación requieren los siguientes pasos:

1. Seleccione la opción Utilizar el modo SDI en Windows en la página "Interfaz" de la caja de diálogo de las Propiedades.
2. Crear una aplicación fusionada (monopuesto y/o aplicación cliente).

Entonces, cuando se ejecute en un contexto soportado (ver arriba), la aplicación fusionada funcionará automáticamente en modo SDI.

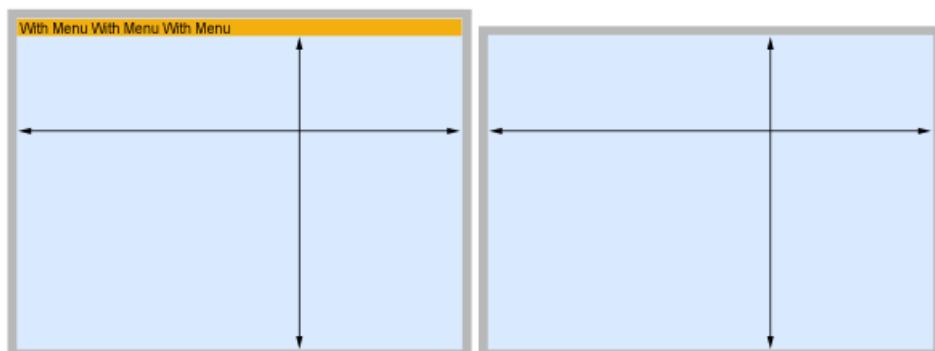
## Gestión de aplicaciones en modo SDI

La ejecución de una aplicación 4D en modo SDI no requiere ninguna implementación específica: las barras de menú existentes se desplazan automáticamente en las propias ventanas SDI. Sin embargo, debe prestar atención a los principios específicos que se enumeran a continuación.

### Menús en las ventanas

En modo SDI, la barra de menú del proceso se muestra automáticamente en cada ventana de tipo documento abierta durante la vida del proceso (esto excluye, por ejemplo, las paletas flotantes). Sin embargo, cuando la barra de menús del proceso no está visible, los accesos directos a los elementos del menú permanecen activos.

Los menús se añaden sobre las ventanas sin modificar el tamaño de su contenido:



Así, las ventanas pueden utilizarse en los modos MDI o SDI sin tener que recalcular la posición de los objetos.

## Sobre la pantalla de inicio

- Si se seleccionó la opción de interfaz Pantalla de bienvenida en las Propiedades, la ventana de bienvenida contendrá los menús que se habrían mostrado en la ventana MDI. Tenga en cuenta también que al cerrar la ventana de la pantalla de inicio se saldrá de la aplicación, al igual que en el modo MDI.
- Si no se ha seleccionado la opción de pantalla de bienvenida, los menús se mostrarán sólo en las ventanas abiertas, según las elecciones del desarrollador.

## Salida automática

Cuando se ejecuta en modo MDI, una aplicación 4D simplemente se cierra cuando el usuario cierra la ventana de la aplicación (ventana MDI). Sin embargo, cuando se ejecutan en modo SDI, las aplicaciones 4D no tienen una ventana de aplicación y, por otra parte, el cierre de la última ventana abierta no significa necesariamente que el usuario quiera que la aplicación salga (pueden estar ejecutándose procesos sin interfaz, por ejemplo) - aunque podría ser lo que se desea.

Para manejar este caso, las aplicaciones 4D ejecutadas en modo SDI incluyen un mecanismo para salir automáticamente (llamando al comando `QUIT 4D`) cuando se cumplen las siguientes condiciones:

- el usuario no puede seguir interactuando con la aplicación
- no hay procesos de usuario en curso
- Los procesos 4D o workers están esperando un evento
- el servidor web no se ha lanzado.

Cuando se llama a un menú con una acción estándar asociada `salir`, la aplicación sale y se cierran todas las ventanas, sea cual sea el lugar desde el que se llamó al menú.

## Lenguaje

Aunque es manejado de forma transparente por 4D, el modo SDI introduce pequeñas variaciones en la gestión de la interfaz de la aplicación. A continuación se enumeran las especificidades del lenguaje 4D.

Comando/funcionalidad	Especificidad en el modo SDI en Windows
<code>Open form window</code>	Opciones para soportar las ventanas flotantes en SDI ( <code>Controller form window</code> ) y para eliminar la barra de menú ( <code>Form has no menu bar</code> )
<code>Menu bar height</code>	Devuelve la altura en píxeles de una línea de barra de menú única, incluso si la barra de menú se ha envuelto en dos o más líneas. Devuelve 0 cuando el comando es llamado desde un proceso sin ventana formulario
<code>SHOW MENU BAR / HIDE MENU BAR</code>	Se aplica sólo a la ventana formulario actual (desde donde se ejecuta el código)
<code>MAXIMIZE WINDOW</code>	La ventana se maximiza al tamaño de la pantalla
<code>CONVERT COORDINATES</code>	<code>XY Screen</code> es el sistema de coordenadas global donde la pantalla principal se sitúa en (0,0). Las pantallas situadas a su izquierda o arriba pueden tener coordenadas negativas y las situadas a su derecha o debajo pueden tener coordenadas mayores que los valores devueltos por <code>Screen height</code> o <code>Screen width</code> .
<code>GET MOUSE</code>	Las coordenadas globales son relativas a la pantalla
<code>GET WINDOW RECT</code>	Cuando se pasa -1 en el parámetro ventana, el comando devuelve 0;0;0;0
<code>On Drop database method</code>	No soportado

# Propiedades usuario

4D provides two modes of operation for project Settings:

- Standard mode: all [settings](#) are stored in the [settings.4DSettings file at the project level](#) and are applied in all cases. This is the default mode, suitable for development phase (all applications).
- User settings mode: part of the custom settings are stored in a [settings.4DSettings file in the Settings folder](#) (for all data files) or [in the Data folder](#) (for this data file) and are used instead of the structure settings. This mode is suitable for deployment phase for Desktop applications. You enable this mode using an option located on the [Security page](#) of the Settings.

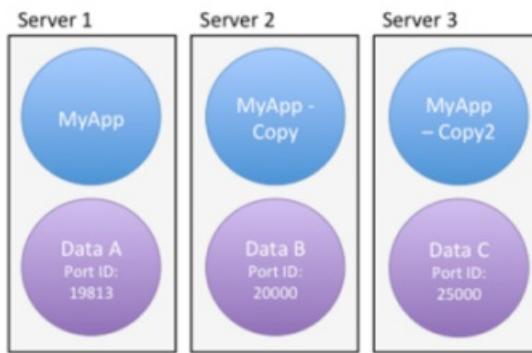
By defining user settings, you can keep custom settings between updates of your 4D applications, or manage different settings for the same 4D application deployed on several different sites. It also makes it possible to use programming to manage setting files using XML.

4D can generate and use two types of user settings:

- User Settings (standard): They are used instead of structure settings for any data file opened with the application.
- User Settings for Data file: They can be defined specifically for each data file used with your application, configuring for example the port ID or the server cache.

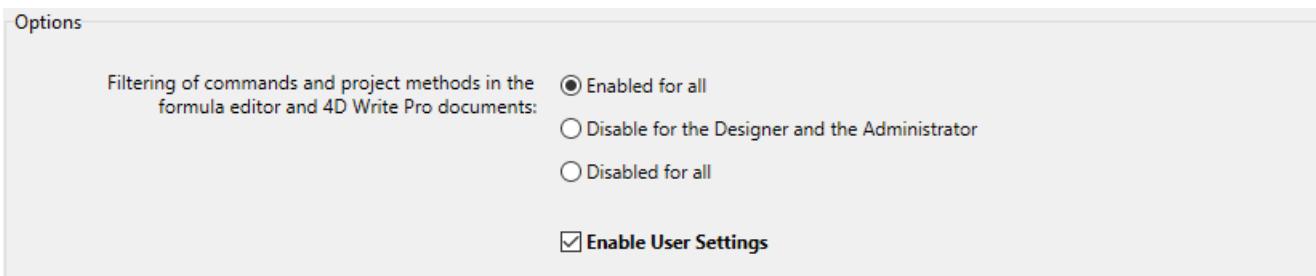
With this option, you can easily deploy and update several copies of the same desktop application with several data files, each containing different settings.

Consider for example the following configuration, where an application is duplicated and each copy uses a different Port ID setting. If this user setting is linked to the data file, you will be able to update the application without having to manually change the Port ID:



## Activar las propiedades usuario

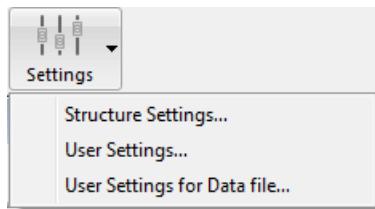
To enable user settings, you need to check the [Settings > Security > Enable User Settings](#) option:



When you check this option, the settings are separated into three dialog boxes:

- Propiedades estructura
- Propiedades usuario
- Propiedades usuario para el archivo de datos

You can access these dialog boxes using the [Design > Settings...](#) menu or the [Settings](#) button in the toolbar:

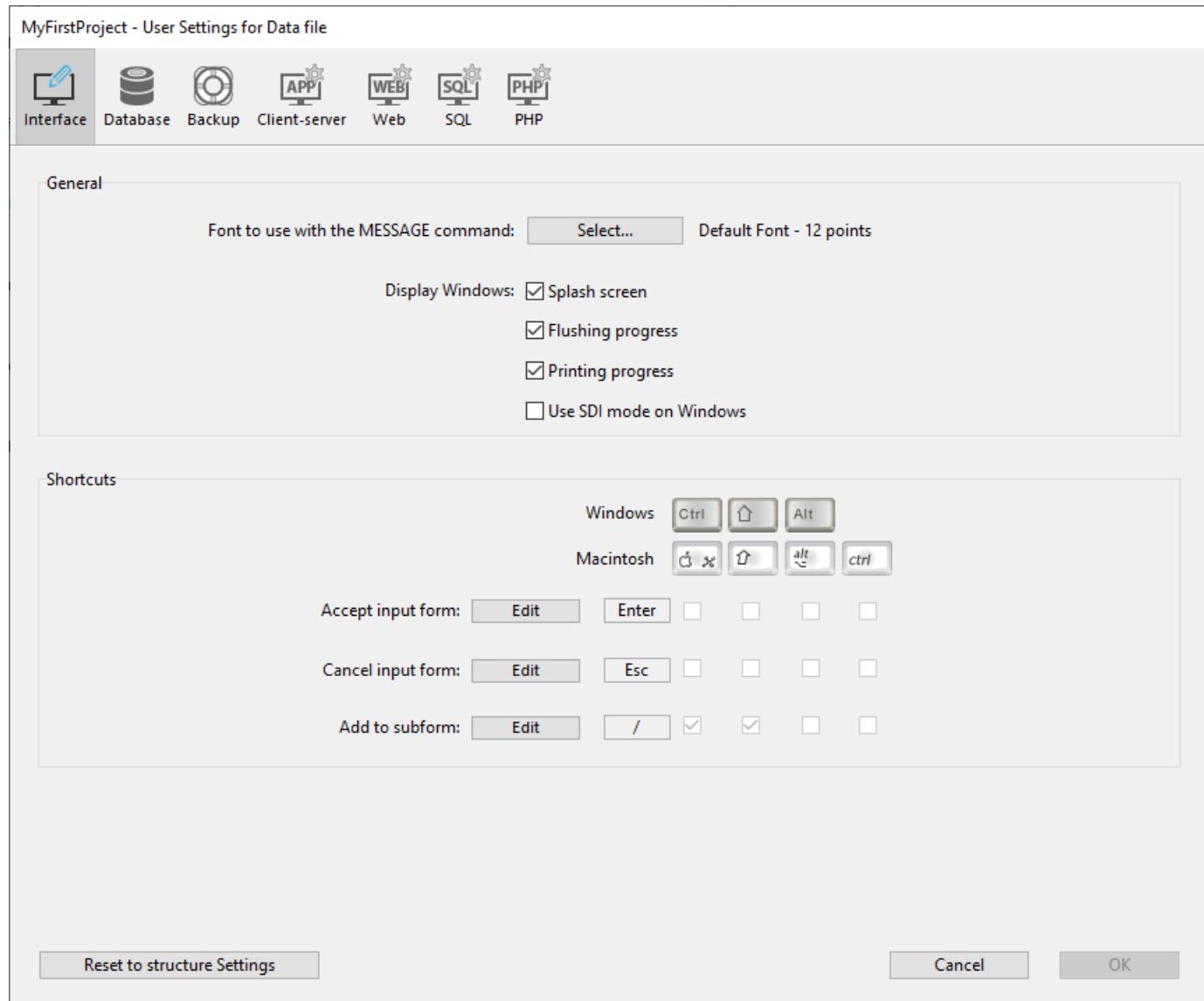


You can also access these dialog boxes using the [OPEN SETTINGS WINDOW](#) command with the appropriate *settingsType* selector.

The Structure Settings dialog box is identical to the standard Settings, and provides access to all its properties (which can be overridden by user settings).

## Propiedades usuario y propiedades de usuario para el archivo de datos

The User Settings and User Settings for Data File dialog boxes contain a selection of relevant properties that can be defined for all data files or a single data file:



The following table lists the pages of settings found in the User Settings and User Settings for Data File dialog boxes and describes their main differences with respect to standard settings:

Página de Propiedades estructura	Página de las Propiedades usuario	Página de Propiedades usuario para archivo de datos
Página General	N/a	N/a
Página interfaz	Idéntica a las Propiedades estándar	Idéntica a las Propiedades estándar
Página Compilador	N/a	N/a
Página Base de datos/almacenamiento de datos	N/a	N/a
Página Base de datos/Memoria	Idéntica a las Propiedades estándar	Idéntica a las Propiedades estándar
Page Backup/Periodicidad	N/a	Idéntica a las Propiedades estándar
Página Backup/Configuración	N/a	Idéntica a las Propiedades estándar
Página Backup & Backup y restaurar	N/a	Idéntica a las Propiedades estándar
Client-server/Network options page	Idéntica a las Propiedades estándar	Idéntica a las Propiedades estándar
Página Cliente-Servidor/Configuración IP	Idéntica a las Propiedades estándar	Idéntica a las Propiedades estándar
Página Web/Configuración	Idéntica a las Propiedades estándar	Idéntica a las Propiedades estándar
Página Web/Opciones (I)	Idéntica a las Propiedades estándar	Idéntica a las Propiedades estándar
Página Web/Opciones (II)	Idéntica a las Propiedades estándar	Idéntica a las Propiedades estándar
Página Web/Log (tipo)	Idéntica a las Propiedades estándar	Idéntica a las Propiedades estándar
Página Web/Log (backup)	Idéntica a las Propiedades estándar	Idéntica a las Propiedades estándar
Página Web/Web services	Opción de prefijación de los métodos no disponible	Opción de prefijación de los métodos no disponible
Página SQL	Idéntica a las Propiedades estándar	Idéntica a las Propiedades estándar
Página PHP	Idéntica a las Propiedades estándar	Idéntica a las Propiedades estándar
Página seguridad	N/a	N/a
Página de compatibilidad	N/a	N/a

When you edit settings in this dialog box, they are automatically stored in the corresponding `settings.4DSettings` file (see below).

## SET DATABASE PARAMETER and user settings

Some of the user settings are also available through the [SET DATABASE PARAMETER](#) command. User settings are parameters with the Kept between two sessions property set to Yes.

When the User Settings feature is enabled, user settings edited by the [SET DATABASE PARAMETER](#) command are automatically saved in the user settings for the data file.

`Table sequence number` is an exception; this setting value is always saved in the data file itself.

## Archivos settings.4DSettings

When you [check the Enable User Settings option](#), user settings files are automatically created. Their location depends on the type of user settings defined.

## Parámetros usuario (estándar)

The standard user settings file is automatically created and placed in a settings folder at the following location:

`ProjectFolder/Settings/settings.4DSettings`

... where *ProjectFolder* is the name of the folder containing the project structure file.

In merged applications, the user settings file is placed at the following location:

- In single-user versions: ProjectFolder/Database/Settings/settings.4DSettings
- In client/server versions: ProjectFolder/Server Database/Settings/settings.4DSettings

## Propiedades usuario para archivo de datos

The user settings file linked to the data file is automatically created and placed in a settings folder at the following location:

`Data/Settings/settings.4DSettings`

... where *Data* is the name of the folder containing the current data file of the application.

When the data file is located at the same level as the project structure file, structure-based and data-based user settings files share the same location and file. The User Settings for Data File... menu command is not proposed.

Settings files are XML files; they can be read and modified using integrated 4D XML commands or using an XML editor. This means that you can manage settings by programming, particularly in the context of applications compiled and merged with 4D Volume Desktop. When you modify this file by programming, the changes are only taken into account the next time the database is opened.

## Prioridad de los parámetros

Las propiedades pueden almacenarse en tres niveles. Each setting defined at one level overrides the same setting defined at a previous level, if any:

Nivel de prioridad	Nombre	Ubicación	Comentarios
3 (el más bajo)	Structure settings (or Settings when "User settings" feature not enabled)	<i>settings.4DSettings</i> file in the Sources folder (project databases) or in the Settings folder as the same level as the structure file (binary databases)	Unique location when user settings are not enabled. Se aplica a todas las copias de la aplicación.
2	Propiedades usuario (todos los archivos de datos)	<i>settings.4DSettings</i> file in the Settings folder at the same level as the Project folder	Reemplaza las propiedades de estructura. Se almacena en el paquete de la aplicación.
1 (el mayor)	Propiedades usuario (archivo de datos actual)	<i>settings.4DSettings</i> file in the Settings folder at the same level as the data file	Reemplaza las propiedades de estructura y las propiedades usuario. Applied only when the linked data file is used with the application.

Keep in mind that user settings files only contain a subset of relevant settings, while the structure file contains all custom settings, including core settings.

# Generador de aplicaciones

4D incluye un generador de aplicaciones para crear un paquete de proyecto (versión final). Este generador simplifica el proceso de finalización y despliegue de las aplicaciones compiladas en 4D. Maneja automáticamente las funcionalidades específicas de los distintos sistemas operativos y facilita el despliegue de aplicaciones cliente-servidor.

El generador de aplicaciones le permite:

- Generar una estructura compilada, sin código interpretado,
- Generar una aplicación autónoma ejecutable, es decir, fusionada con 4D Volume Desktop, el motor de base de datos 4D,
- Generar diferentes aplicaciones a partir de la misma estructura compilada mediante un proyecto XML,
- Generar aplicaciones cliente-servidor homogéneas,
- Generar aplicaciones cliente-servidor con actualización automática de los componentes del cliente y del servidor.
- Guardar sus parámetros de generación para su uso futuro (botón *Guardar los parámetros*).

Las aplicaciones compiladas se basan en archivos **.4dz** que son de sólo lectura. Tenga en cuenta que el uso de comandos o funciones que modifican los archivos fuente (como `CREATE INDEX` o `CREATE TABLE` (SQL)) no es posible por defecto en las aplicaciones compiladas. Sin embargo, puede crear aplicaciones específicas que soporten modificaciones locales utilizando la llave XML `PackProject` (ver [doc.4d.com](#)).

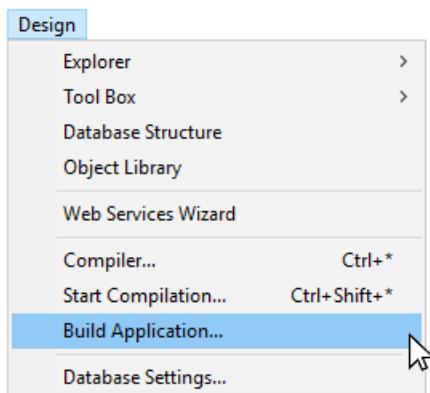
## Generalidades

Generar un paquete de proyecto puede realizarse utilizando:

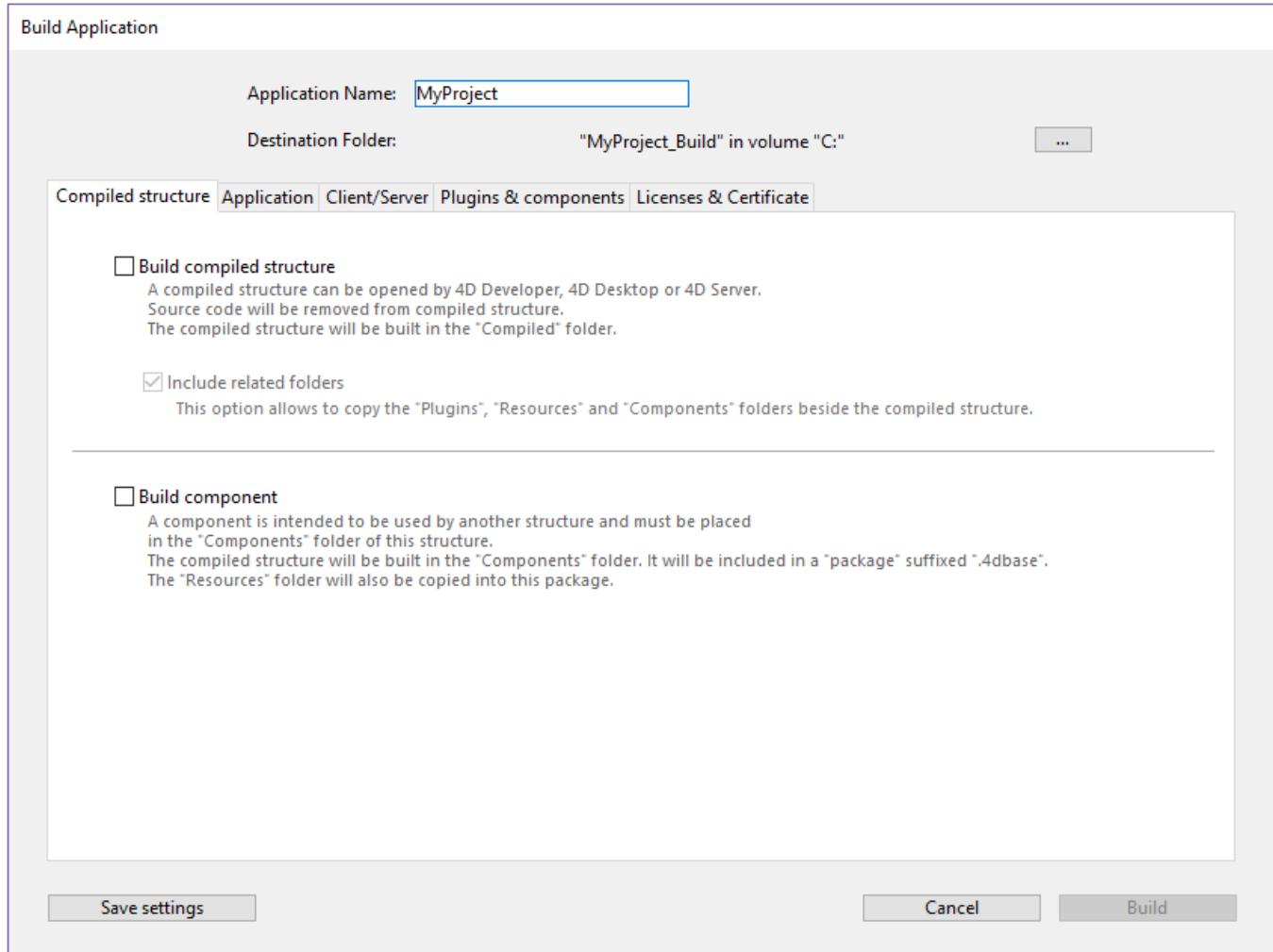
- el comando **BUILD APPLICATION**,
- or the [Build Application dialog](#).

## Diálogo crear aplicación

To display the Build application dialog, select Design > Build Application... from the menu bar.



La caja de diálogo del generador de aplicaciones incluye varias páginas a las que se puede acceder mediante pestañas:



La generación sólo puede efectuarse una vez compilado el proyecto. Si selecciona este comando sin haber compilado previamente el proyecto, o si el código compilado no se corresponde con el código interpretado, aparece una caja de diálogo de advertencia que indica que el proyecto debe ser (re)compilado.

## Parámetros del generador de aplicaciones

Cada parámetro de generación de la aplicación se almacena como una llave XML en el archivo proyecto de la aplicación llamada `"buildApp.4DSettings"`, ubicado en la carpeta `Settings` del proyecto.

Los parámetros por defecto se utilizan la primera vez que se utiliza la caja de diálogo del Generador de aplicaciones. El contenido del archivo proyecto se actualiza, si es necesario, al hacer clic en Construir o Guardar los parámetros. Puede definir varios archivos de parámetros XML para el mismo proyecto y utilizarlos con el comando [BUILD APPLICATION](#).

Las llaves XML ofrecen opciones adicionales a las que se muestran en la caja de diálogo del Generador de aplicaciones. La descripción de estas llaves se detalla en el manual [4D XML Keys BuildApplication](#).

## Archivo de historial

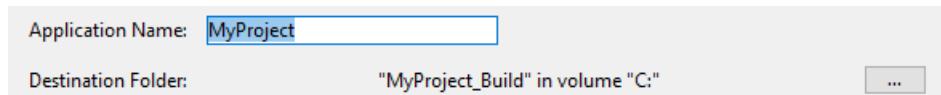
Cuando se crea una aplicación, 4D genera un archivo de registro llamado `BuildApp.log.xml` en la carpeta `Logs` del proyecto. El archivo de historial almacena la siguiente información para cada generación:

- El inicio y el fin de la generación de objetivos,
- El nombre y la ruta de acceso completa de los archivos generados,
- La fecha y la hora de la generación,
- Todos los errores que se han producido,
- Todo problema de firma (por ejemplo, un plug-in no firmado).

La verificación de este archivo puede ayudarle a ahorrar tiempo durante los siguientes pasos de despliegue, por ejemplo, si tiene la intención de notarizar su aplicación.

Utilice el comando `Get 4D file(Build application log file)` para obtener la ubicación del archivo de registro.

## Nombre de la aplicación y carpeta de destino



Introduzca el nombre de la aplicación en Nombre de la aplicación.

Especifique la carpeta para la aplicación generada en la Carpeta de destino. Si la carpeta especificada no existe todavía, 4D creará una carpeta *Build*.

## Página de estructura compilada

Esta pestaña le permite generar un archivo de estructura compilado estándar y/o un componente compilado:

A screenshot of the 'Build Application' dialog box, specifically the 'Compiled structure' tab. It shows the same 'Application Name:' and 'Destination Folder:' fields as the previous screenshot. Below these, there are several configuration options:

- Build compiled structure**  
A compiled structure can be opened by 4D Developer, 4D Desktop or 4D Server.  
Source code will be removed from compiled structure.  
The compiled structure will be built in the "Compiled" folder.
- Include related folders**  
This option allows to copy the "Plugins", "Resources" and "Components" folders beside the compiled structure.
- Build component**  
A component is intended to be used by another structure and must be placed in the "Components" folder of this structure.  
The compiled structure will be built in the "Components" folder. It will be included in a "package" suffixed ".4dbase".  
The "Resources" folder will also be copied into this package.

At the bottom of the dialog are three buttons: 'Save settings', 'Cancel', and 'Build'.

## Generar una estructura compilada

Genera una aplicación que sólo contiene código compilado.

Esta funcionalidad crea un archivo `.4dz` en una carpeta `Compiled Database/<project name>`. Por ejemplo, si ha llamado a su aplicación "MyProject", 4D creará:

`<destination>/Compiled Database/MyProject/MyProject.4dz`

Un archivo `.4dz` es esencialmente una versión comprimida (empaquetada) de la carpeta del proyecto. Un archivo `.4dz` es esencialmente una versión comprimida (empaquetada) de la carpeta del proyecto. El tamaño compacto y optimizado

de los archivos .4dz hace que los paquetes de proyectos sean fáciles de desplegar.

When generating .4dz files, 4D uses a standard zip format by default. The advantage of this format is that it is easily readable by any unzip tool. If you do not want to use this standard format, add the `UseStandardZipFormat` XML key with value `False` in your `buildApp.4DSettings` file (for more information, see the *4D XML Keys Backup* manual on [doc.4d.com](http://doc.4d.com)).

## Incluir las carpetas asociadas

Cuando se marca esta opción, todas las carpetas relacionadas con el proyecto se copian en la carpeta Build como carpetas *Components* y *Resources*. Para más información sobre estas carpetas, consulte la [descripción de la arquitectura del proyecto](#).

## Generar un componente

Genera un componente compilado a partir de la estructura.

Un componente es un proyecto estándar 4D en el que se han desarrollado funcionalidades específicas. Un componente es un proyecto estándar 4D en el que se han desarrollado funcionalidades específicas.

Si ha llamado a su aplicación, *MiComponente*, 4D creará una carpeta *Components* que contiene la carpeta *MiComponente.4dbase*:

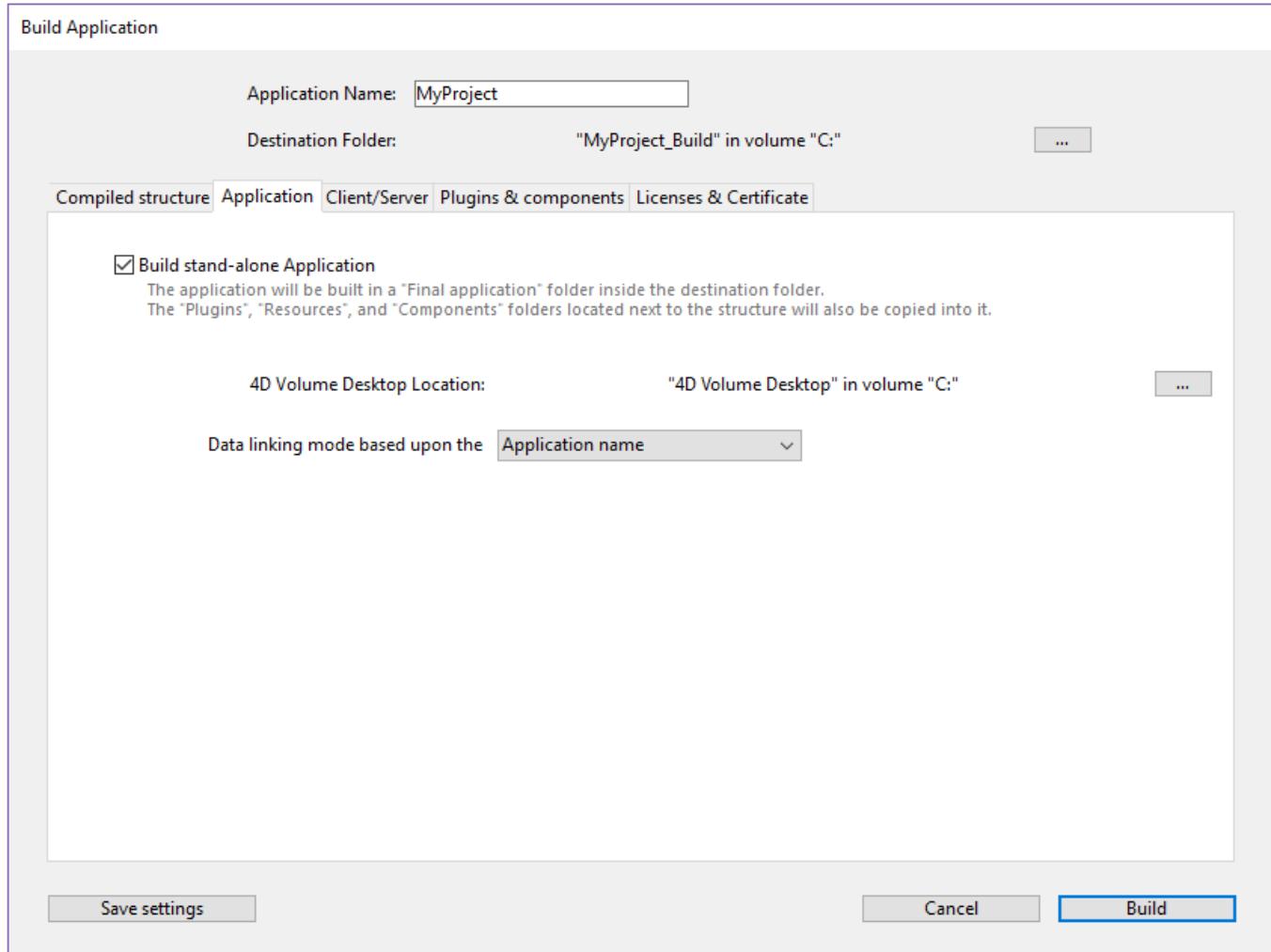
`<destination>/Components/MyComponent.4dbase/MyComponent.4DZ`.

La carpeta *MyComponent.4dbase* contiene:

- archivo *MyComponent.4DZ*
- Una carpeta *Resources*: todos los resources asociados se copian automáticamente en esta carpeta. Los otros componentes y/o carpetas de plugins no se copian (un componente no puede utilizar plugins u otros componentes).

## Página Application

Esta pestaña le permite crear una versión autónoma y monopuesto de su aplicación:



## Crear una aplicación autónoma

Al marcar la opción Crear una aplicación autónoma y hacer clic en Generar se creará una aplicación autónoma (con doble clic) directamente desde su proyecto de aplicación.

Los siguientes elementos son necesarios para la creación:

- 4D Volume Desktop (el motor de la base de datos 4D),
- una [licencia apropiada](#)

En Windows, esta función crea un archivo ejecutable (.exe). En macOS, se encarga de la creación de paquetes de software.

El principio consiste en fusionar un archivo de estructura compilado con 4D Volume Desktop. Las funcionalidades ofrecidas por el archivo 4D Volume Desktop están relacionadas con la oferta de productos a la que se ha suscrito. Las funcionalidades ofrecidas por el archivo 4D Volume Desktop están relacionadas con la oferta de productos a la que se ha suscrito.

Puede definir un archivo de datos por defecto o permitir a los usuarios crear y utilizar su propio archivo de datos (ver la sección [Gestión de archivos de datos en las aplicaciones finales](#)).

Es posible automatizar la actualización de las aplicaciones monopuesto fusionadas mediante una secuencia de comandos de lenguaje (ver [Actualización automática de aplicaciones servidor o monopuesto](#)).

### Ubicación de 4D Volume Desktop

Para crear una aplicación autónoma, primero debe designar la carpeta que contiene el archivo 4D Volume Desktop:

- *Windows* - la carpeta contiene los archivos 4D Volume Desktop.4DE, 4D Volume Desktop. RSR, así como varios archivos y carpetas necesarios para su funcionamiento. Estos elementos deben colocarse al mismo nivel que la carpeta seleccionada.
- *macOS* - 4D Volume Desktop se entrega en forma de un paquete de software estructurado que contiene varios

archivos y carpetas genéricos.

Para seleccionar la carpeta 4D Volume Desktop, haga clic en el botón [...]. Aparece una caja de diálogo que le permite designar la carpeta de 4D Volume Desktop (Windows) o el paquete (macOS).

Una vez seleccionada la carpeta, se muestra su ruta completa y, si realmente contiene 4D Volume Desktop, se activa la opción de generación de una aplicación ejecutable.

El número de versión de 4D Volume Desktop debe coincidir con el número de versión de 4D Developer Edition. Por ejemplo, si utiliza 4D Developer v18, debe seleccionar un 4D Volume Desktop v18.

## Modo de enlace de datos

Esta opción permite elegir el modo de enlace entre la aplicación fusionada y el archivo de datos local. Hay dos modos de enlazar disponibles:

- Por nombre de la aplicación (por defecto) - La aplicación 4D abre automáticamente el archivo de datos abierto más recientemente correspondiente al archivo de estructura. Esto le permite mover el paquete de aplicaciones libremente en el disco. Esta opción debería usarse generalmente para aplicaciones fusionadas, a menos que necesite específicamente duplicar su aplicación.
- Por ruta de la aplicación - La aplicación 4D fusionada analizará el archivo *lastDataPath.xml* de la aplicación e intentará abrir el archivo de datos con un atributo "executablePath" que coincida con la ruta completa de la aplicación. Si se encuentra una entrada de este tipo, se abre su correspondiente archivo de datos (definido a través de su atributo "dataFilePath"). Si se encuentra una entrada de este tipo, se abre su correspondiente archivo de datos (definido a través de su atributo "dataFilePath").

Para más información sobre el modo de vinculación de datos, consulte la sección [Último archivo de datos abierto](#).

## Archivos generados

Al hacer clic en el botón Generar, 4D crea automáticamente una carpeta Final Application en la carpeta de destino definida. Dentro de la carpeta Final Application hay una subcarpeta con el nombre de la aplicación especificada.

Si ha especificado "MyProject" como nombre de la aplicación, encontrará los siguientes archivos en esta subcarpeta (MyProject):

- *Windows*
  - MyProject.exe - Su ejecutable y un MyProject.rsr (los recursos de la aplicación)
  - Las carpetas 4D Extensions y Resources, varias librerías (DLL), la carpeta Native Components y SASL Plugins - Archivos necesarios para el funcionamiento de la aplicación
  - Una carpeta Database - Incluye una carpeta Resources y un archivo MyProject.4DZ. Constituyen la estructura compilada del proyecto, así como también la carpeta Resources. Nota: esta carpeta también contiene la carpeta *Default Data*, si se ha definido (ver [Gestión de archivos de datos en las aplicaciones finales](#)).
  - (Opcional) Carpeta de componentes y/o carpeta Plugins - Contiene todos los componentes y/o archivos de plugins incluidos en el proyecto. Para más información sobre este punto, consulte la sección [Plugins y componentes](#).
  - Carpeta Licenses - Un archivo XML de números de licencia integrados en la aplicación. Para obtener más información sobre este punto, consulte la sección [Licencias y certificados](#).
  - Elementos adicionales añadidos a la carpeta 4D Volume Desktop, si los hay (ver [Personalizar la carpeta 4D Volume Desktop](#)).

Todos estos elementos deben estar en la misma carpeta para que el ejecutable funcione.

- *macOS*
  - Un paquete de software llamado MyProject.app que contiene su aplicación y todos los elementos necesarios para su funcionamiento, incluyendo los plug-ins, componentes y licencias. Para más información sobre la integración de plug-ins y componentes, consulte la sección [\[Plugins y componentes\]\(#plugins-and-components\)](#). Para obtener más información sobre la integración de licencias, consulte la sección [\[Licencias y certificados\]\(#licenses-and-certificate\)](#). Nota: en macOS, el comando [Archivo aplicación](#) del lenguaje 4D devuelve la ruta del archivo NombreApplication (situado en la carpeta Contents:macOS del paquete de software) y no la del archivo .comp (carpeta Contents).

## Personalizar la carpeta 4D Volume Desktop

Cuando se construye una aplicación independiente, 4D copia el contenido de la carpeta 4D Volume Desktop en la carpeta Destination > *Final Application*. A continuación, podrá personalizar el contenido de la carpeta 4D Volume Desktop original según sus necesidades. Puede, por ejemplo:

- Instalar una versión de 4D Volume Desktop correspondiente a un lenguaje específico;
- Añadir una carpeta *PlugIns* personalizada;
- Personalizar el contenido de la carpeta *Resources*.

The macOS packages built contain the same items as the Windows subfolders. Puede visualizar su contenido (Control+clic en el ícono) para poder modificarlo.

## Ubicación de los archivos web

Si su aplicación ejecutable se utiliza como servidor web, los archivos y los archivos y carpetas requeridos por el servidor deben instalarse en ubicaciones específicas. Estos elementos son los siguientes:

- archivos *cert.pem* y *key.pem* (opcional): estos archivos se utilizan para las conexiones TLS y por los comandos de encriptación de datos,
- carpeta raíz web por defecto.

Los elementos deben ser instalados:

- En Windows: en la subcarpeta *Final Application\MyProject\Database*.
- En macOS: junto al paquete de software *MyProject.app*.

## Página Cliente/Servidor

On this tab, you can build customized client-server applications that are homogenous, cross-platform and with an automatic update option.

**Build Application**

Application Name:	<input type="text" value="myapp"/>
Destination Folder:	"myapp_build" in volume "C:" <input type="button" value="..."/>
<input type="button" value="Compiled structure"/> <input type="button" value="Application"/> <input type="button" value="Client/Server"/> <input type="button" value="Plugins &amp; components"/> <input type="button" value="Licenses &amp; Certificate"/>	
<p><input type="checkbox"/> <b>Build server application</b> The server will be built for the current platform and can only be launched on it. Macintosh and Windows clients will be able to connect to it.</p> <p><b>4D Server location:</b></p> <p>Current version: <input type="text" value="1"/> Data linking mode based upon the <input type="button" value="Application name"/></p> <p><input type="checkbox"/> Allow connection of Silicon macOS clients</p> <p>Compiled structure location: <input type="button" value="..."/></p> <hr/> <p><input type="checkbox"/> <b>Build client application</b> The client will be built for the current platform and can only be launched on it. It can connect to a server running on either Macintosh or Windows.</p> <p><b>4D Volume Desktop Location:</b></p> <p><b>Copy of client applications inside the server application</b> This operation allows the server to automatically send the client applications each time they are updated.</p> <p><input type="checkbox"/> Allow automatic update of Windows client application</p> <p><input type="checkbox"/> Allow automatic update of Macintosh client application</p> <p>macOS client update archive location: <input type="button" value="..."/></p>	
<input type="button" value="Save settings"/> <input type="button" value="Cancel"/> <input type="button" value="Build"/>	

## ¿Qué es una aplicación cliente/servidor?

A client/server application comes from the combination of three items:

- Un proyecto 4D compilado,
- La aplicación 4D Server,
- La aplicación 4D Volumen Desktop (macOS y/o Windows).

Once built, a client/server application is composed of two customized parts: the Server portion (unique) and the Client portion (to install on each client machine).

If you want to deploy a client/server application in an heterogeneous environment (client applications running on Intel/AMD and Apple Silicon machines), it is recommended to [compile the project for all processors](#) on a macOS machine, so that all client applications will run natively.

Also, the client/server application is customized and its handling simplified:

- Para lanzar la parte del servidor, el usuario simplemente hace doble clic en la aplicación servidor. No es necesario seleccionar el archivo proyecto.
- Para lanzar la parte cliente, el usuario simplemente hace doble clic en la aplicación cliente, que se conecta directamente a la aplicación servidor. No es necesario elegir un servidor en una caja de diálogo de conexión. Si desea que la aplicación cliente se conecte al servidor utilizando una dirección específica (distinta del nombre del servidor publicado en la subred), debe utilizar la llave XML `IPAddress` en el archivo `buildapp.4DSettings`. Si la conexión falla, [se pueden implementar mecanismos alternativos específicos](#management-of-client-connections). Puede "forzar" la visualización de la caja de diálogo de conexión estándar presionando la tecla Opción (macOS) o Alt (Windows) mientras inicia la aplicación cliente. Sólo la parte cliente puede conectarse a la parte del servidor correspondiente. Si un usuario intenta conectarse a la parte servidor utilizando una aplicación estándar 4D, se devuelve un mensaje de error y la conexión es imposible.
- Una aplicación cliente/servidor puede configurarse para que la parte cliente [se actualice automáticamente a través de la red](#). Sólo es necesario crear y distribuir una versión inicial de la aplicación cliente, las actualizaciones

posteriores se gestionan mediante el mecanismo de actualización automática.

- También es posible automatizar la actualización de la parte del servidor mediante el uso de una secuencia de comandos del lenguaje ([SET UPDATE FOLDER](#) y [RESTART 4D](#)).

## Construir aplicación servidor

Check this option to generate the server part of your application during the building phase. Debe designar la ubicación en su disco de la aplicación 4D Server que va a utilizar. Debe designar la ubicación en su disco de la aplicación 4D Server que va a utilizar.

### Ubicación de 4D Server

Click on the [...] button and use the *Browse for folder* dialog box to locate the 4D Server application. In macOS, you must select the 4D Server package directly.

### Versión actual

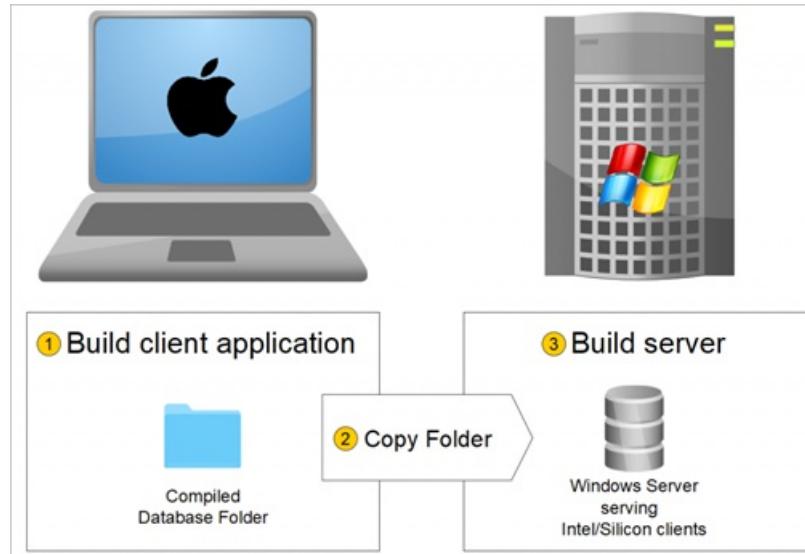
Used to indicate the current version number for the application generated. A continuación, podrá aceptar o rechazar las conexiones de las aplicaciones cliente en función de su número de versión. El intervalo de compatibilidad de las aplicaciones del cliente y del servidor se define mediante el uso de [llaves XML](#) específicas).

### Autorizar la conexión de los clientes Silicon Mac

When building a server on Windows, check this option to allow Apple Silicon clients to connect to your server application. You can then specify a path to the structure compiled for Apple Silicon/Intel.

To allow Apple Silicon clients to connect to a Server application built on Windows, you must first build a client application on macOS, with a project compiled for Apple Silicon and Intel. This automatically creates a compiled structure, identical to the one created with the [Build compiled structure](#) option (without the related folders).

Then, you can copy that structure to your Windows machine, and use it to build the server application:



### Compiled structure location

Path to compiled structure of the Apple Silicon/Intel client application used to build a Windows Server (see [Allow connection of Silicon Mac clients](#)).

### Modo de enlace de datos

Esta opción permite elegir el modo de enlace entre la aplicación fusionada y el archivo de datos local. Hay dos modos de enlazar disponibles:

- Por nombre de la aplicación (por defecto) - La aplicación 4D abre automáticamente el archivo de datos abierto más recientemente correspondiente al archivo de estructura. Esto le permite mover el paquete de aplicaciones libremente en el disco. Esta opción debería usarse generalmente para aplicaciones fusionadas, a menos que necesite específicamente duplicar su aplicación.

- Por ruta de la aplicación - La aplicación 4D fusionada analizará el archivo *lastDataPath.xml* de la aplicación e intentará abrir el archivo de datos con un atributo "executablePath" que coincida con la ruta completa de la aplicación. Si se encuentra una entrada de este tipo, se abre su correspondiente archivo de datos (definido a través de su atributo "dataFilePath"). Si se encuentra una entrada de este tipo, se abre su correspondiente archivo de datos (definido a través de su atributo "dataFilePath").

Para más información sobre el modo de vinculación de datos, consulte la sección [Último archivo de datos abierto](#).

## Construir la aplicación cliente

Checking this option generates the client part of your application during the building phase.

Puede marcar esta opción:

- along with the [Build server application](#) option to build matching server and client parts for the current platform and (optionally) include the automatic update archive files,
- without selecting the [Build server application](#) option, usually to build the update archive file to be selected from the "concurrent" platform when building the server part.

### Ubicación de 4D Volume Desktop

Designates the location on your disk of the 4D Volume Desktop application to be used to build the client part of your application.

El número de versión de 4D Volume Desktop debe coincidir con el número de versión de 4D Developer Edition.  
El número de versión de 4D Volume Desktop debe coincidir con el número de versión de 4D Developer Edition.

The 4D Volume Desktop must correspond to the current platform (which will also be the platform of the client application). If you want to build a client application for the "concurrent" platform, you must carry out an additional build operation using a 4D application running on that platform.

If you want the client application to connect to the server using a specific address (other than the server name published on the sub-network), you must use the `IPAddress` XML key in the `buildapp.4DSettings` file. Para más información sobre este archivo, consulte la descripción del comando [BUILD APPLICATION](#). También puede implementar mecanismos específicos en caso de fallo de la conexión. También puede implementar mecanismos específicos en caso de fallo de la conexión.

### Copy of client applications inside the server application

The options of this area set up the mechanism for updating the client part(s) of your client/server applications using the network each time a new version of the application is generated. These options are only enabled when the Build client application option is checked.

- Allow automatic update of Windows client application - Check this option to build a `.4darchive` file that can be sent to your client applications on the Windows platform in case of update.
- Allow automatic update of Macintosh client application - Check this option to build a `.4darchive` file that can be sent to your client applications on the Macintosh platform in case of update.

The `.4darchive` is copied at the following location:

```
<ApplicationName>_Build/Client Server executable/Upgrade4DClient/
```

### Selecting client archive for the concurrent platform

You can check the Allow automatic update... option for client applications running on the concurrent platform. Esta opción sólo se activa si:

- the Build server application option is checked,
- the Allow automatic update... option for client applications running on the current platform is checked.

This feature requires that you click on the [...] button and designate the location on your disk of the file to use for the

update. The file to select depends on the current server platform:

Plataforma del servidor actual	Archivo requerido	Detalles
macOS	Windows 4D Volume Desktop o Windows client update archive	By default, you select the <code>4D Volume Desktop</code> application for Windows. To select a <code>.4darchive</code> file previously built on Windows, press Shift while clicking on [...]
Windows	macOS client update archive	Select a signed <code>.4darchive</code> file previously built on macOS

You can build specific a `.4darchive` file on the concurrent platform by selecting only the [Build client application](#) and the appropriate [Allow automatic update...](#) option.

## Visualización de la notificación de actualización

The client application update notification is carried out automatically following the server application update.

It works as follows: when a new version of the client/server application is built using the application builder, the new client portion is copied as a compressed file in the Upgrade4DCClient subfolder of the ApplicationName Server folder (in macOS, these folders are included in the server package). If you have followed the process for generating a cross-platform client application, a `.4darchive` update file is available for each platform:

To trigger client application update notifications, simply replace the old version of the server application with the new one and then execute it. El resto del proceso es automático.

On the client side, when the “old” client application tries to connect to the updated server application, a dialog box is displayed on the client machine, indicating that a new version is available. The user can either update their version or cancel the dialog box.

- If the user clicks OK, the new version is downloaded to the client machine over the network. Una vez finalizada la descarga, se cierra la aplicación cliente antigua y se lanza la nueva versión, que se conecta al servidor. Una vez finalizada la descarga, se cierra la aplicación cliente antigua y se lanza la nueva versión, que se conecta al servidor.
- If the user clicks Cancel, the update is cancelled; if the old version of the client application is not in the range of versions accepted by the server (please refer to the following paragraph), the application is closed and connection is impossible. Otherwise (by default), the connection is established.

## Forzar las actualizaciones automáticas

In some cases, you may want to prevent client applications from being able to cancel the update download. For example, if you used a new version of the 4D Server source application, the new version of the client application must absolutely be installed on each client machine.

To force the update, simply exclude the current version number of client applications (X-1 and earlier) in the version number range compatible with the server application. En este caso, el mecanismo de actualización no permitirá que las aplicaciones cliente no actualizadas se conecten. En este caso, el mecanismo de actualización no permitirá que las aplicaciones cliente no actualizadas se conecten.

The [current version number](#) is set on the Client/Server page of the Build Application dialog box. The intervals of authorized numbers are set in the application project using specific [XML keys](#).

## En caso de error

If 4D cannot carry out the update of the client application, the client machine displays the following error message: “The update of the client application failed. La aplicación va a cerrar ahora.”

Hay muchas causas posibles para este error. When you get this message, it is advisable to check the following parameters first off:

- Pathnames - Check the validity of the pathnames set in the application project via the Application builder dialog box or via XML keys (for example `ClientMacFolderToWin`). More particularly, check the pathnames to the versions of 4D Volume Desktop.

- Read/write privileges - On the client machine, check that the current user has write access rights for the client application update.

## Archivos generados

Once a client/server application is built, you will find a new folder in the destination folder named `Client Server executable`. This folder contains two subfolders, `<ApplicationName>Client` and `<ApplicationName>Server`.

These folders are not generated if an error occurs. In this case, open the [log file](#) in order to find out the cause of the error.

The `<ApplicationName>Client` folder contains the client portion of the application corresponding to the execution platform of the application builder. Esta carpeta debe instalarse en cada máquina cliente. The `<ApplicationName>Server` folder contains the server portion of the application.

The contents of these folders vary depending on the current platform:

- *Windows* - Each folder contains the application executable file, named `<ApplicationName>Client.exe` for the client part and `<ApplicationName>Server.exe` for the server part as well as the corresponding `.rsr` files. The folders also contain various files and folders necessary for the applications to work and customized items that may be in the original 4D Volume Desktop and 4D Server folders.
- *macOS* - Each folder contains only the application package, named `<ApplicationName> Cliente` para la parte cliente y `<ApplicationName> Server` para la parte del servidor. Cada paquete contiene todos los elementos necesarios para que la aplicación funcione. Under macOS, launch a package by double-clicking it.

The macOS packages built contain the same items as the Windows subfolders. Para modificarlo, primero hay que mostrar su contenido (Control+clic en el ícono).

If you checked the "Allow automatic update of client application" option, an additional subfolder called `Upgrade4DClient` is added in the `<ApplicationName>Server` folder/package. Esta subcarpeta contiene la aplicación cliente en formato macOS y/o Windows como archivo comprimido. Esta subcarpeta contiene la aplicación cliente en formato macOS y/o Windows como archivo comprimido.

## Ubicación de los archivos web

If the server and/or client part of your double-clickable application is used as a Web server, the files and folders required by the server must be installed in specific locations. Estos elementos son los siguientes:

- archivos `cert.pem` y `key.pem` (opcional): estos archivos se utilizan para las conexiones SSL y por los comandos de encriptación de datos,
- Carpeta raíz web por defecto (WebFolder).

Los elementos deben ser instalados:

- en Windows
  - Server application - in the `Client Server executable`\ `<ApplicationName>Server\Server Database` subfolder.
  - Client application - in the `Client Server executable`\ `<ApplicationName>Client` subfolder.
- en macOS
  - Server application - next to the `<ApplicationName>Server` software package.
  - Client application - next to the `<ApplicationName>Client` software package.

## Integrar una aplicación cliente monopuesto

4D allows you to embed a compiled structure in the Client application. This feature can be used, for example, to provide users with a "portal" application, that gives access to different server applications thanks to the `OPEN DATABASE` command executing a `.4dlink` file.

To enable this feature, add the `DatabaseToEmbedInClientWinFolder` and/or `DatabaseToEmbedInClientMacFolder` keys

in the `buildApp` settings file. When one of these keys is present, the client application building process generates a single-user application: the compiled structure, instead of the `EnginedServer.4Dlink` file, is placed in the "Database" folder.

- If a default data folder exists in the single-user application, a licence is embedded.
- If no default data folder exists in the single-user application, it will be executed without data file and without licence.

El escenario básico es:

1. In the Build application dialog box, select the "Build compiled structure" option to produce a .4DZ or .4DC for the application to be used in single-user mode.
2. In the `buildApp.4DSettings` file of the client-server application, use following xml key(s) to indicate the path to the folder containing the compiled single user application:
  - `DatabaseToEmbedInClientWinFolder`
  - `DatabaseToEmbedInClientMacFolder`
3. Genere la aplicación cliente-servidor. Esto tendrá los siguientes efectos:
  - the whole folder of the single user application is copied inside the "Database" folder of the merged client
  - the `EnginedServer.4Dlink` file of the "Database" folder is not generated
  - the .4DC, .4DZ, .4DIndy files of the single user application copy are renamed using the name of the merged client
  - the `PublishName` key is not copied in the `info.plist` of the merged client
  - if the single-user application does not have a "Default data" folder, the merged client will run with no data.

Automatic update 4D Server features ([Current version](#) number, `SET UPDATE FOLDER` command...) work with single-user application as with standard remote application. At connection, the single-user application compares its `CurrentVers` key to the 4D Server version range. If outside the range, the updated client application is downloaded from the server and the Updater launches the local update process.

## Customizing client and/or server cache folder names

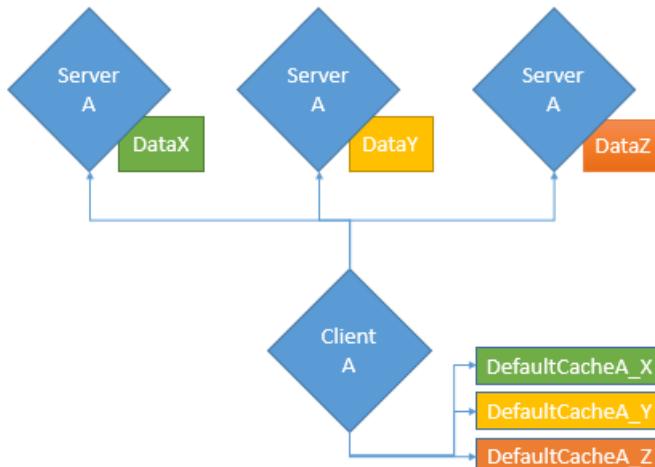
Client and server cache folders are used to store shared elements such as resources or components. They are required to manage exchanges between server and remote clients. Client/server applications use default pathnames for both client and server system cache folders.

In some specific cases, you might need to customize the names of these folders to implement specific architectures (see below). 4D provides you with the `ClientServerSystemFolderName` and `ServerStructureFolderName` keys to be set in the `buildApp` settings file.

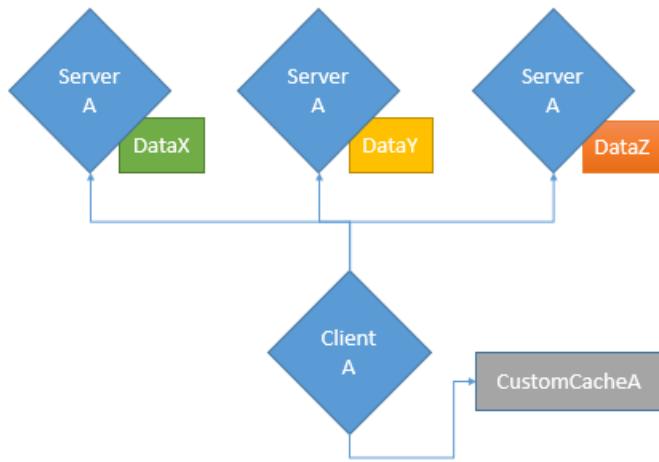
### Carpeta de caché cliente

Customizing the client-side cache folder name can be useful when your client application is used to connect to several merged servers which are similar but use different data sets. In this case, to save multiple unnecessary downloads of identical local resources, you can use the same custom local cache folder.

- Default configuration (for each connection to a server, a specific cache folder is downloaded/updated ):



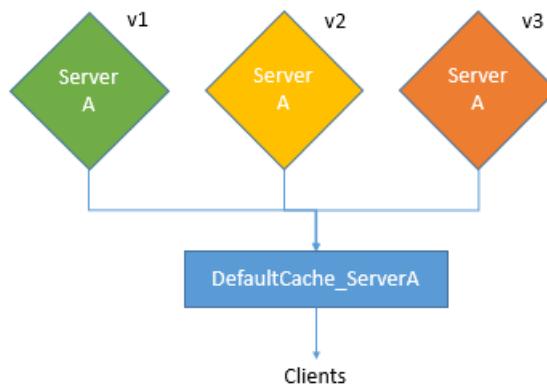
- Using the `ClientServerSystemFolderName` key (a single cache folder is used for all servers):



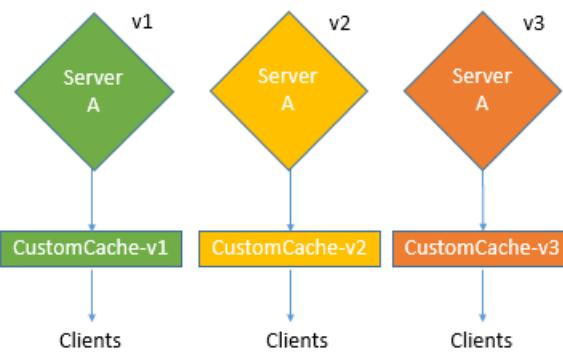
## Carpeta de caché del servidor

Customizing the server-side cache folder name is useful when you run several identical server applications built with different 4D versions on the same computer. If you want each server to use its own set of resources, you need to customize the server cache folder.

- Default configuration (*same server applications share the same cache folder*):



- Using the `ServerStructureFolderName` key (*a dedicated cache folder is used for each server application*):



## Página Plugins y componentes

On this tab, you set each **plug-in** and each **component** that you will use in your stand-alone or client/server application.

The page lists the elements loaded by the current 4D application:



- Active column - Indicates that the items will be integrated into the application package built. Todos los elementos están marcados por defecto. To exclude a plug-in or a component, deselect the check box next to it.
- Plugins and components column - Displays the name of the plug-in/component.
- ID column - Displays the plug-in/component's identification number (if any).
- Type column - Indicates the type of item: plug-in or component.

If you want to integrate other plug-ins or components into the executable application, you just need to place them in a PlugIns or Components folder next to the 4D Volume Desktop application or next to the 4D Server application. The mechanism for copying the contents of the source application folder (see [Customizing the 4D Volume Desktop folder](#)) can be used to integrate any type of file into the executable application.

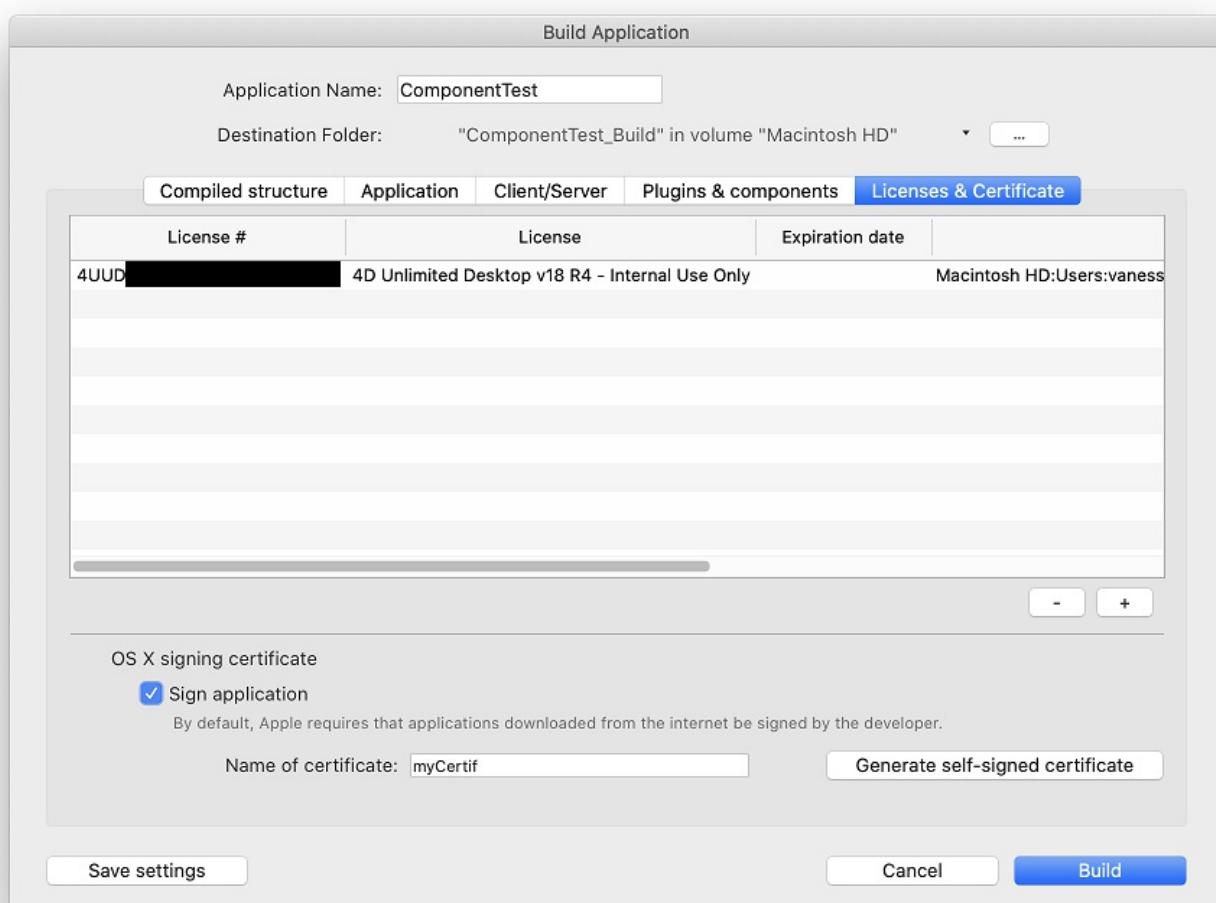
If there is a conflict between two different versions of the same plug-in (one loaded by 4D and the other located in the source application folder), priority goes to the plug-in installed in the 4D Volume Desktop/4D Server folder. However, if there are two instances of the same component, the application will not open.

The use of plug-ins and/or components in a deployment version requires the necessary license numbers.

## Página Licences & Certificado

The Licences & Certificate page can be used to:

- designate the license number(s) that you want to integrate into your single-user stand-alone application
- sign the application by means of a certificate in macOS.



## Licencias

This tab displays the list of available deployment licenses that you can integrate into your application. Por defecto, la lista está vacía. Debe añadir explícitamente su licencia *4D Developer Professional*, así como cada licencia *4D Desktop Volume* que se vaya a utilizar en la aplicación generada. You can add another 4D Developer Professional number and its associated licenses other than the one currently being used.

To remove or add a license, use the [+] and [-] buttons at the bottom of the window.

When you click on the [+] button, an open file dialog box appears displaying by default the contents of the *Licenses* folder of your machine. For more information about the location of this folder, refer to the [Get 4D folder](#) command.

You must designate the files that contain your Developer license as well as those containing your deployment licenses. These files were generated or updated when the *4D Developer Professional* license and the *4D Desktop Volume* licenses were purchased.

Once you have selected a file, the list will indicate the characteristics of the license that it contains.

- License # - Product license number
- License - Name of the product
- Expiration date - Expiration date of the license (if any)
- Ruta de acceso - Ubicación en el disco

If a license is not valid, a message will warn you.

Puede designar tantos archivos válidos como desee. When building an executable application, 4D will use the most appropriate license available.

Dedicated "R" licenses are required to build applications based upon "R-release" versions (license numbers for "R" products start with "R-4DDP").

After the application is built, a new deployment license file is automatically included in the Licenses folder next to the executable application (Windows) or in the package (macOS).

## Certificación de las aplicaciones en OS X

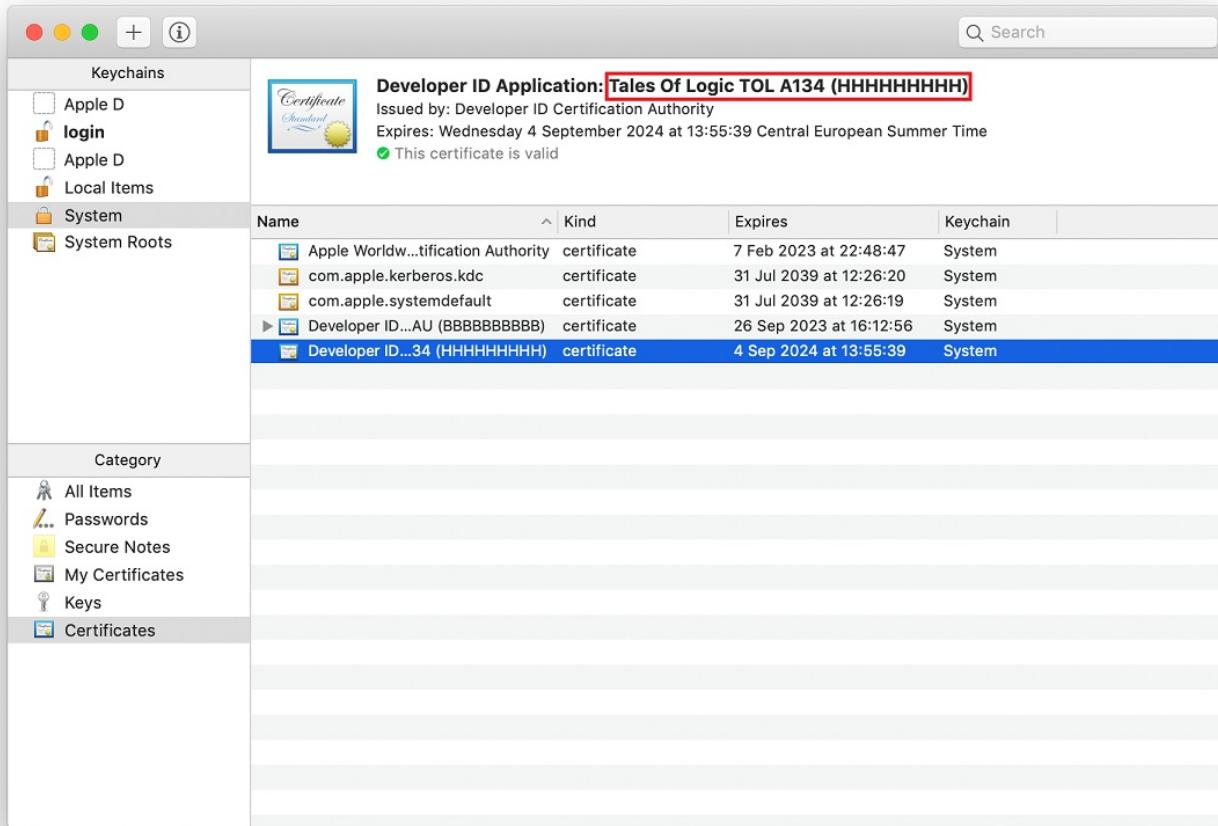
El generador de aplicaciones puede firmar aplicaciones 4D fusionadas bajo macOS (aplicaciones monopuesto, componentes, 4D Server y partes cliente bajo macOS). Signing an application authorizes it to be executed using the Gatekeeper functionality of macOS when the "Mac App Store and identified Developers" option is selected (see "About Gatekeeper" below).

- Check the Sign application option to include certification in the application builder procedure for OS X. 4D will check the availability of elements required for certification when the build occurs:

The screenshot shows a dialog box titled "OS X signing certificate". It contains a checked checkbox labeled "Sign application" and a note stating "By default, Apple requires that applications downloaded from the internet be signed by the developer." Below the checkbox is a text input field labeled "Name of certificate:" followed by a "Generate self-signed certificate" button.

This option is displayed under both Windows and macOS, but it is only taken into account for macOS versions.

- Name of certificate - Enter the name of your developer certificate validated by Apple in this entry area. The certificate name is usually the name of the certificate in the Keychain Access utility (part in red in the following example):

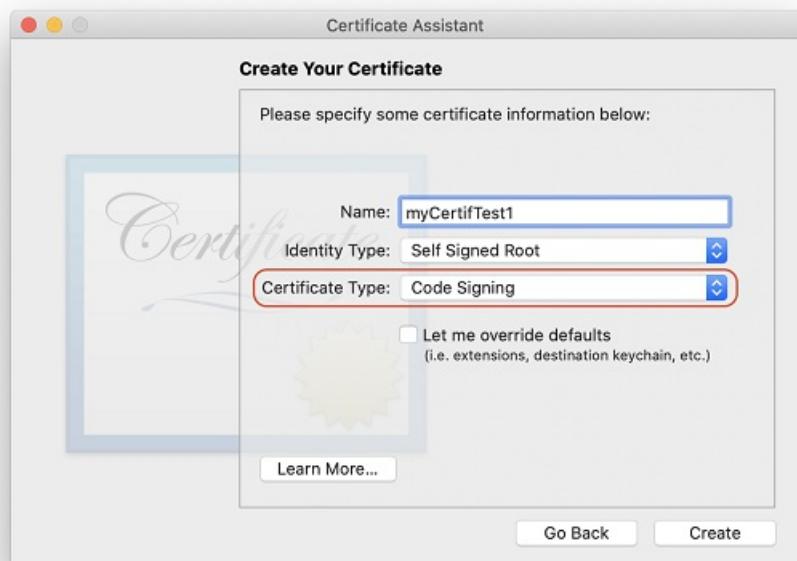


To obtain a developer certificate from Apple, Inc., you can use the commands of the Keychain Access menu or go here: <http://developer.apple.com/library/mac/#documentation/Security/Conceptual/CodeSigningGuide/Procedures/Procedures.html>.

This certificate requires the presence of the Apple codesign utility, which is provided by default and usually located in the "/usr/bin/" folder. If an error occurs, make sure that this utility is present on your disk.

- Generate self-signed certificate - runs the "Certificate Assistant" that allows you to generate a self-signed certificate. If you do not have an Apple developer certificate, you need to provide a self-signed certificate. With this certificate, no alert message is displayed if the application is deployed internally. If the application is deployed externally (i.e. through http or email), at launch macOS displays an alert message that the application's developer is unidentified. The user can "force" the opening of the application.

In the "Certificate Assistant", be sure to select the appropriate options:



4D recommends to subscribe to the Apple Developer Program to get access to Developer Certificates that are necessary to notarize applications (see below).

## Sobre Gatekeeper

Gatekeeper is a security feature of OS X that controls the execution of applications downloaded from the Internet. If a downloaded application does not come from the Apple Store or is not signed, it is rejected and cannot be launched.

On Apple Silicon machines, 4D [components](#) need to be actually signed. An unsigned component will generate an error at application startup ("lib4d-arm64.dylib can't be opened...").

La opción Firmar la aplicación del Generador de aplicaciones de 4D le permite generar aplicaciones y componentes compatibles con esta opción por defecto.

## Sobre la notarización

Application notarization is highly recommended by Apple as of macOS 10.14.5 (Mojave) and 10.15 (Catalina), since

non-notarized applications deployed via the internet are blocked by default.

Las [funciones de firma integradas](#) de 4D se han adaptado para cumplir con todos los requisitos de Apple para permitir el uso del servicio de notario de Apple. La notarización en sí debe ser realizada por el desarrollador y es independiente de 4D (tenga en cuenta también que requiere la instalación de Xcode). La notarización en sí debe ser realizada por el desarrollador y es independiente de 4D (tenga en cuenta también que requiere la instalación de Xcode).

For more information on the notarization concept, please refer to [this page on the Apple developer website](#).

## Personalizar los iconos de una aplicación

4D associates a default icon with stand-alone, server, and client applications, however you can customize the icon for each application.

- macOs - When building a double-clickable application, 4D handles the customizing of the icon. In order to do this, you must create an icon file (icns type), prior to building the application file, and place it next to the project folder.

Apple, Inc. provides a specific tool for building *icns* icon files (for more information, please refer to [Apple documentation](#)).

Your icon file must have the same name as the project file and include the *\*.icns\** extension. 4D automatically takes this file into account when building the double-clickable application.

- Windows - When building a double-clickable application, 4D handles the customizing of its icon. In order to do this, you must create an icon file (*.ico* extension), prior to building the application file, and place it next to the project folder.

Your icon file must have the same name as the project file and include the *.ico* extension. 4D automatically takes this file into account when building the double-clickable application.

You can also set specific [XML keys](#) in the `buildApp.4DSettings` file to designate each icon to use. The following keys are available:

- `RuntimeVLIconWinPath`
- `RuntimeVLIconMacPath`
- `ServerIconWinPath`
- `ServerIconMacPath`
- `ClientMacIconForMacPath`
- `ClientWinIconForMacPath`
- `ClientMacIconForWinPath`
- `ClientWinIconForWinPath`

## Gestión de archivos de datos

### Apertura del archivo de datos

When a user launches a merged application or an update (single-user or client/server applications), 4D tries to select a valid data file. Several locations are examined by the application successively.

The opening sequence for launching a merged application is:

- 4D tries to open the last data file opened, [as described below](#) (not applicable during initial launch).
- If not found, 4D tries to open the data file in a default data folder next to the *.4DZ* file in read-only mode.
- If not found, 4D tries to open the standard default data file (same name and same location as the *.4DZ* file).
- If not found, 4D displays a standard "Open data file" dialog box.

### Último archivo de datos abierto

## Ruta del último archivo de datos

Any standalone or server applications built with 4D stores the path of the last data file opened in the application's user preferences folder.

The location of the application's user preferences folder corresponds to the path returned by the following statement:

```
userPrefs:=Get 4D folder(Carpeta 4D activa)
```

The data file path is stored in a dedicated file, named *lastDataPath.xml*.

Thanks to this architecture, when you provide an update of your application, the local user data file (last data file used) is opened automatically at first launch.

This mechanism is usually suitable for standard deployments. However, for specific needs, for example if you duplicate your merged applications, you might want to change the way that the data file is linked to the application (described below).

### Configuring the data linking mode

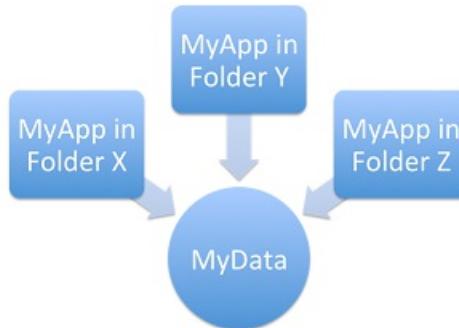
With your compiled applications, 4D automatically uses the last data file opened. By default, the path of the data file is stored in the application's user preferences folder and is linked to the application name.

This may be unsuitable if you want to duplicate a merged application intended to use different data files. Duplicated applications actually share the application's user preferences folder and thus, always use the same data file -- even if the data file is renamed, because the last file used for the application is opened.

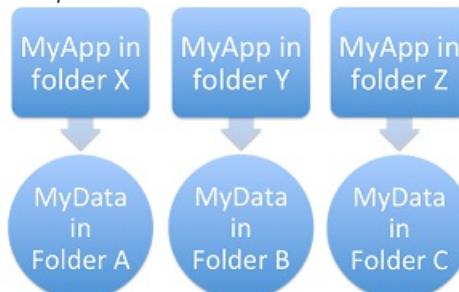
4D therefore lets you link the data file path to the application path. En este caso, el archivo de datos se relacionará con una ruta específica y no será simplemente el último archivo abierto. En este caso, el archivo de datos se relacionará con una ruta específica y no será simplemente el último archivo abierto.

This mode allows you to duplicate your merged applications without breaking the link to the data file. However, with this option, if the application package is moved on the disk, the user will be prompted for a data file, since the application path will no longer match the "executablePath" attribute (after a user has selected a data file, the *lastDataPath.xml* file is updated accordingly).

*Duplication when data linked by application name:*



*Duplication when data linked by application path:*



You can select the data linking mode during the build application process. Puede:

- Use the [Application page](#) or [Client/Server page](#) of the Build Application dialog box.
- Use the LastDataPathLookup XML key (single-user application or server application).

## Definir una carpeta de datos por defecto

4D allows you to define a default data file at the application building stage. Cuando la aplicación se lanza por primera vez, si no se encuentra ningún archivo de datos local (ver [secuencia de lanzamiento descrita anteriormente] (#opening-the-data-file)), el archivo de datos por defecto se abre automáticamente y de forma silenciosa en modo de sólo lectura por 4D. Cuando la aplicación se lanza por primera vez, si no se encuentra ningún archivo de datos local (ver [secuencia de lanzamiento descrita anteriormente] (#opening-the-data-file)), el archivo de datos por defecto se abre automáticamente y de forma silenciosa en modo de sólo lectura por 4D.

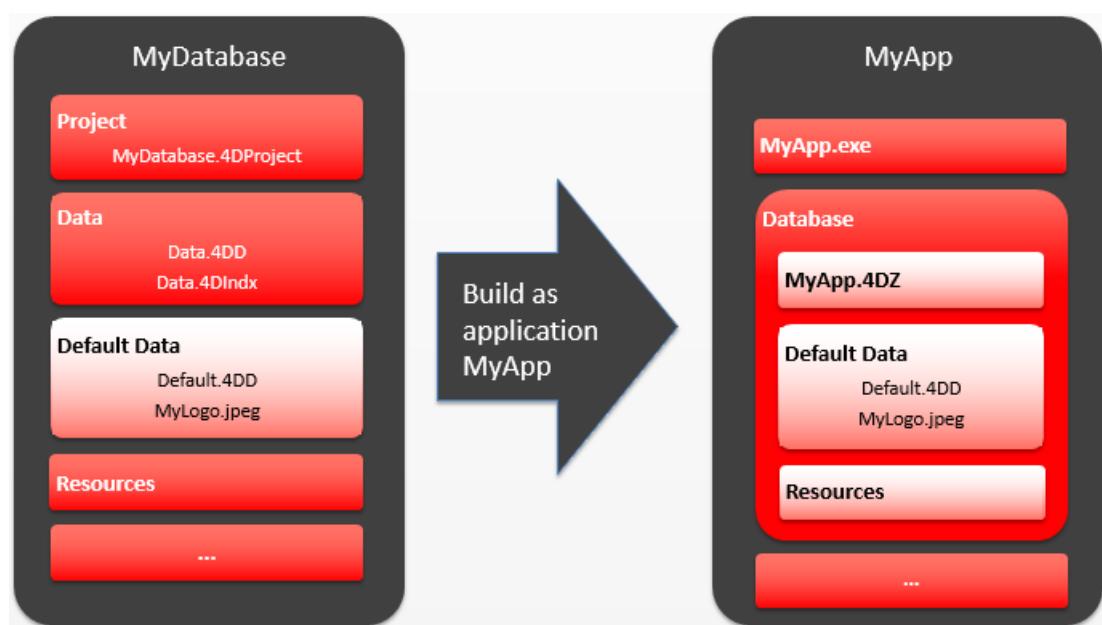
More specifically, the following cases are covered:

- Avoiding the display of the 4D "Open Data File" dialog box when launching a new or updated merged application. You can detect, for example at startup, that the default data file has been opened and thus execute your own code and/or dialogs to create or select a local data file.
- Allowing the distribution of merged applications with read-only data (for demo applications, for instance).

Para definir y utilizar un archivo de datos por defecto:

- You provide a default data file (named "Default.4DD") and store it in a default folder (named "Default Data") inside the application project folder. This file must be provided along with all other necessary files, depending on the project configuration: index (.4DIndx), external Blobs, journal, etc. Es su responsabilidad proveer un archivo de datos válido por defecto. Es su responsabilidad proveer un archivo de datos válido por defecto.
- When the application is built, the default data folder is integrated into the merged application. All files within this default data folder are also embedded.

El siguiente gráfico ilustra esta funcionalidad:



When the default data file is detected at first launch, it is silently opened in read-only mode, thus allowing you to execute any custom operations that do not modify the data file itself.

## Gestión de la conexión(es) de las aplicaciones clientes

The management of connections by client applications covers the mechanisms by which a merged client application connects to the target server, once it is in its production environment.

### Escenario de conexión

The connection procedure for merged client applications supports cases where the dedicated server is not available. The startup scenario for a 4D client application is the following:

1. If valid connection information is stored in the "EnginedServer.4DLink" file within the client application, the client application connects to the specified server address.

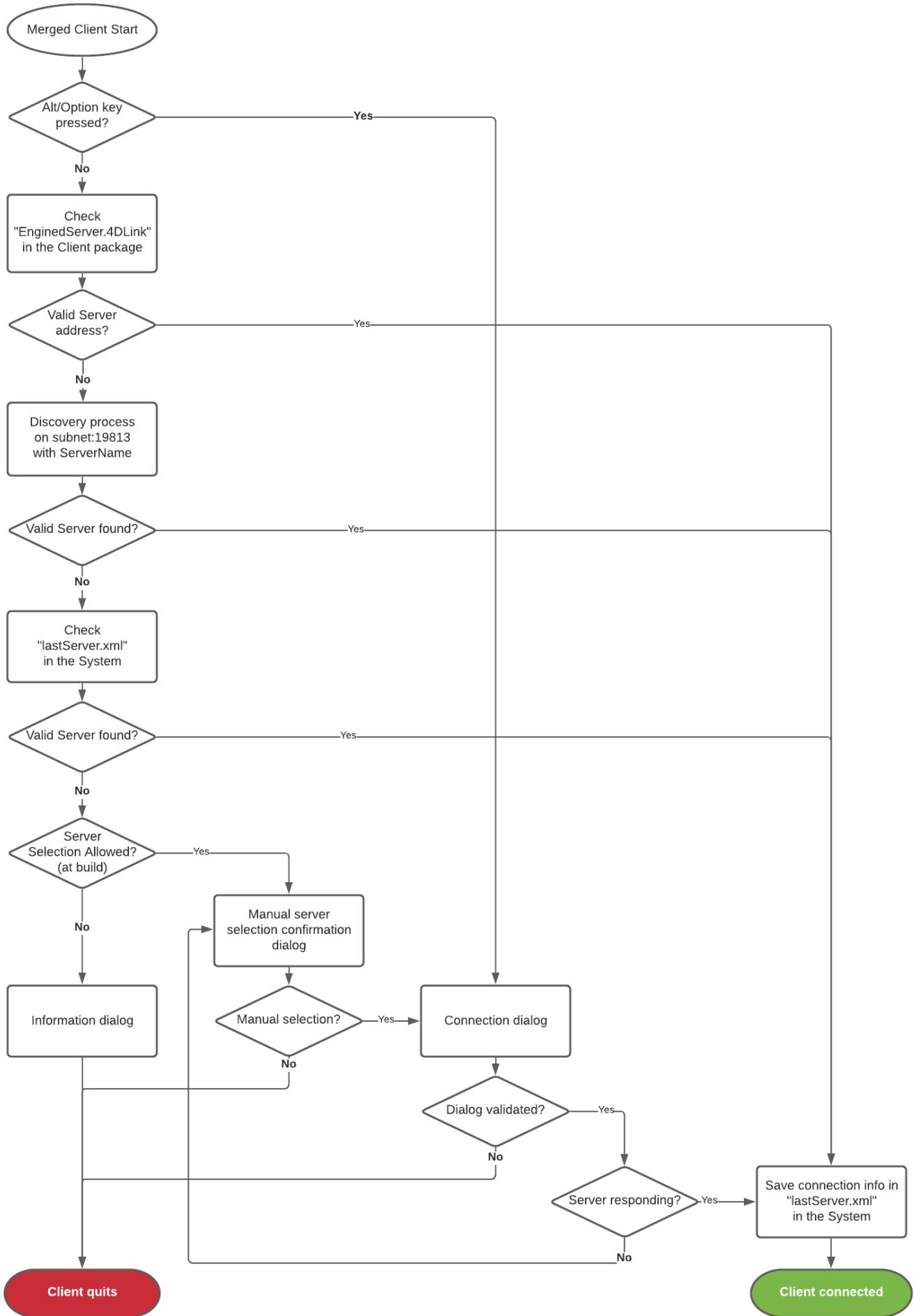
OR

The client application tries to connect to the server using the discovery service (based upon the server name,

broadcasted on the same subnet).

2. If this fails, the client application tries to connect to the server using information stored in the application's user preferences folder ("lastServer.xml" file, see last step).
3. If this fails, the client application displays a connection error dialog box.
  - o If the user clicks on the Select... button (when allowed by the 4D developer at the build step, see below), the standard "Server connection" dialog box is displayed.
  - o If the user clicks on the Quit button, the client application quits.
4. If the connection is successful, the client application saves this connection information in the application's user preferences folder for future use.

The whole procedure is described in the following diagram:



## Almacenando la última ruta del servidor

The last used and validated server path is automatically saved in a file named "lastServer.xml" in the application's user

preferences folder. Esta carpeta se guarda en la siguiente ubicación:

```
userPrefs:=Get 4D folder(Carpeta 4D activa)
```

This mechanism addresses the case where the primary targeted server is temporary unavailable for some reason (maintenance mode for example). Cuando se produce este caso por primera vez, se muestra la caja de diálogo de selección de servidor (si está permitido, ver más adelante) y el usuario puede seleccionar manualmente un servidor alternativo, cuya ruta se guarda si la conexión tiene éxito. Any subsequent unavailability would be handled automatically through the "lastServer.xml" path information.

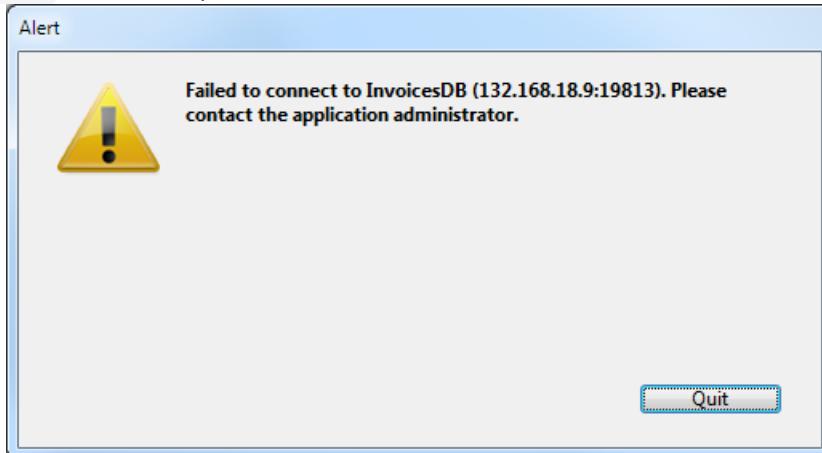
- When client applications cannot permanently benefit from the discovery service, for example because of the network configuration, it is recommended that the developer provide a host name at build time using the [IPAddress](#) key in the "BuildApp.4DSettings" file. El mecanismo aborda los casos de indisponibilidad temporal.
- Pressing the Alt/Option key at startup to display the server selection dialog box is still supported in all cases.

## Availability of the server selection dialog box in case of error

You can choose whether or not to display the standard server selection dialog box on merged client applications when the server cannot be reached. The configuration depends on the value of the [ServerSelectionAllowed](#) XML key on the machine where the application was built:

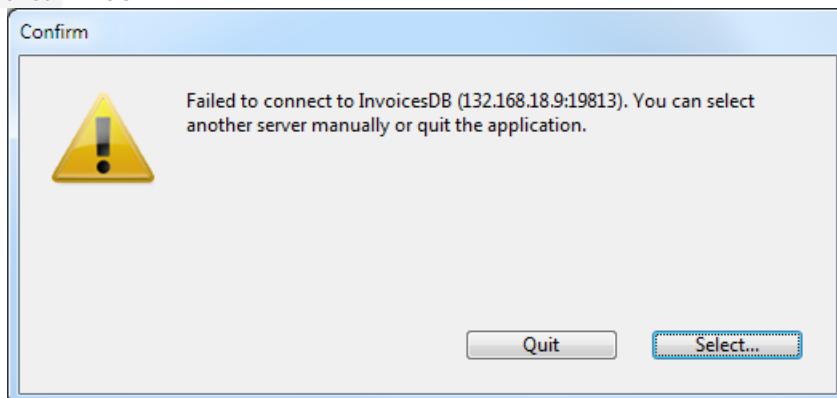
- Display of an error message with no access possible to the server selection dialog box . Funcionamiento por defecto. The application can only quit.

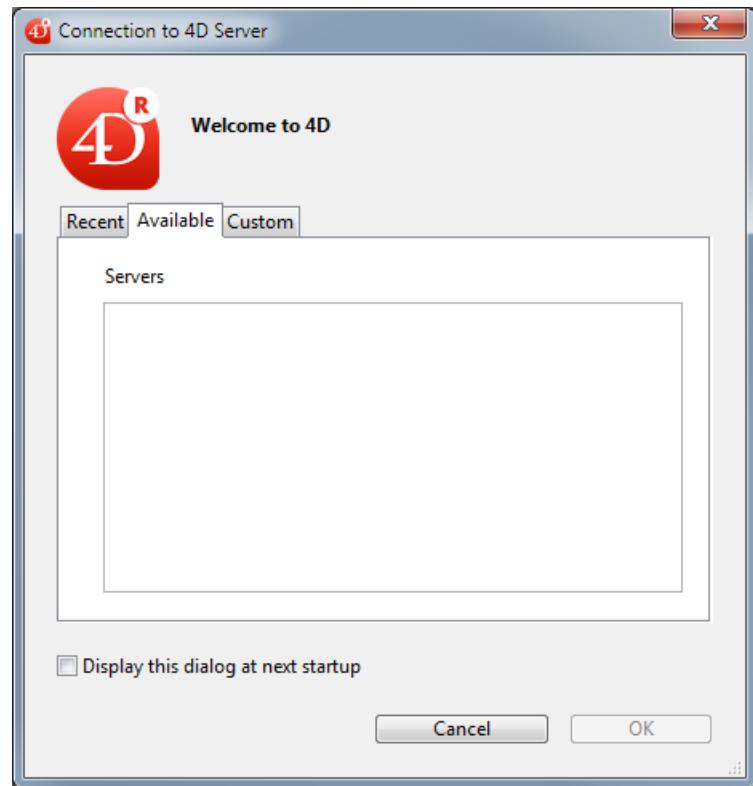
[ServerSelectionAllowed](#) : False or key omitted



- Display of an error message with access to the server selection dialog box possible . The user can access the server selection window by clicking on the Select... button.

[ServerSelectionAllowed](#) : True

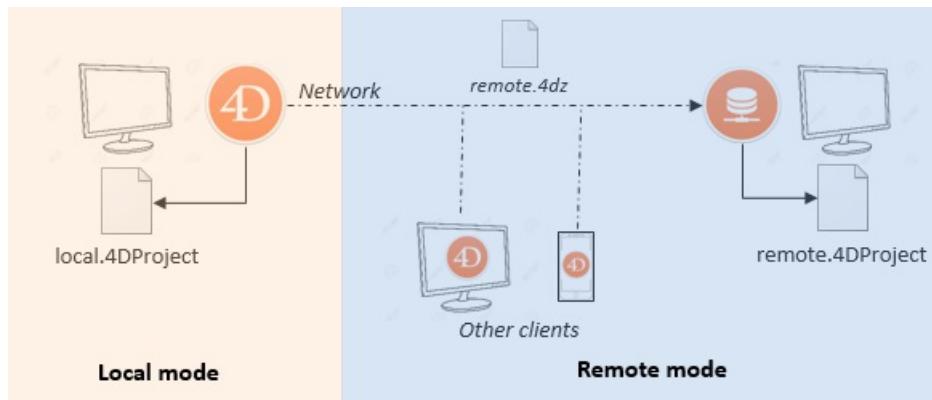




# Gestión Cliente/Servidor

Las aplicaciones 4D Desktop pueden utilizarse en una configuración Cliente/Servidor, ya sea como aplicaciones combinadas cliente/servidor o como proyectos remotos.

- merged client/server applications are generated by the [Build Application manager](#). Se utilizan para el despliegue de aplicaciones.
- remote projects are [.4DProject](#) files opened by 4D Server and accessed with 4D in remote mode. The server sends a .4dz version of the project ([compressed format](#)) to the remote 4D, thus structure files are read-only. This configuration is usually used for application testing.



Connecting to a remote project from the same machine as 4D Server allows modifying the project files. This [specific feature](#) allows to develop a client/server application in the same context as the deployment context.

## Abrir una aplicación cliente/servidor fusionada

A merged client/server application is customized and its starting is simplified:

- Para lanzar la parte del servidor, el usuario simplemente hace doble clic en la aplicación servidor. No es necesario seleccionar el archivo proyecto.
- Para lanzar la parte cliente, el usuario simplemente hace doble clic en la aplicación cliente, que se conecta directamente a la aplicación servidor.

These principles are detailed in the [Build Application](#) page.

## Abrir un proyecto remoto

The first time you connect to a 4D Server project via a remote 4D, you will usually use the standard connection dialog. Thereafter, you will be able to connect directly using the Open Recent Projects menu or a 4DLink shortcut file.

Para conectarse remotamente a un proyecto 4D Server:

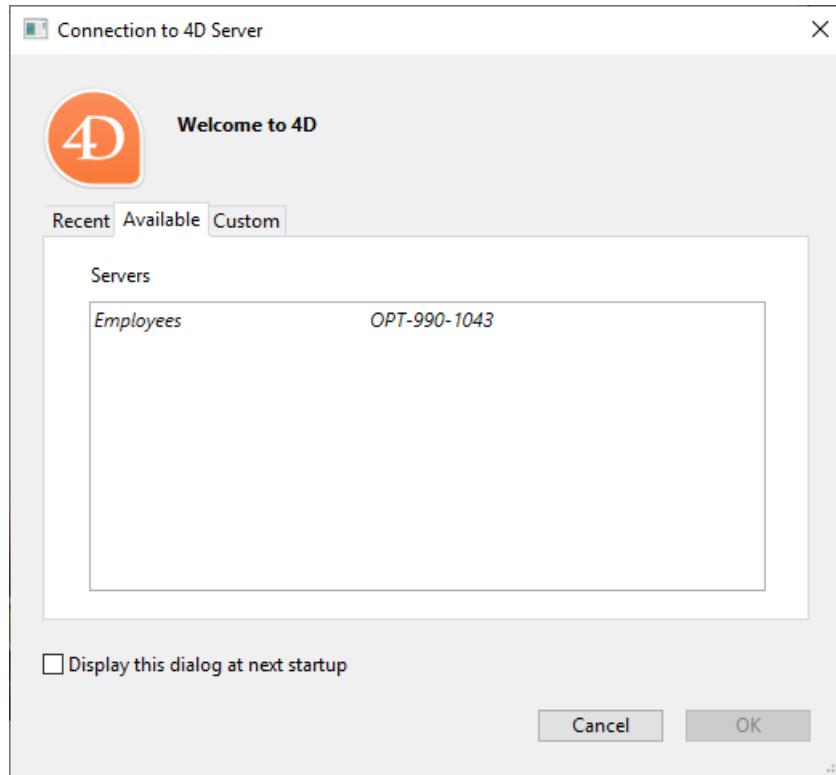
1. Select Connect to 4D Server in the Welcome Wizard dialog,

O

Select Open/Remote Project... from the File menu or the Open toolbar button.

Aparece el diálogo de conexión de 4D Server. This dialog has three tabs: Recent, Available, and Custom.

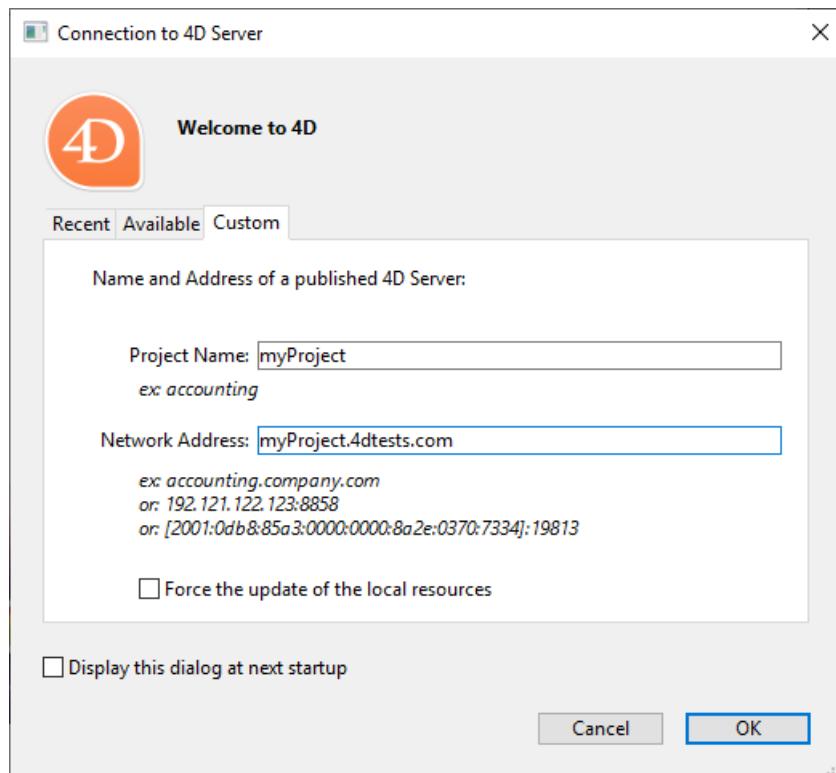
If 4D Server is connected to the same network as the remote 4D, select Available. 4D Server includes a built-in TCP/IP broadcasting system that, by default, publishes the name of the 4D Server projects available over the network. The list is sorted by order of appearance and updated dynamically.



To connect to a server from the list, double-click on its name or select it and click the OK button.

A circumflex accent (^) is placed before the name of projects published with the encryption option enabled.

If the published project is not displayed in the Available list, select Custom. The Custom page allows you to connect to a published server on the network using its network address and assigning it a customized name.



- Project name: Defines the local name of the 4D Server project. This name will be used in the Recent page when referring to the project.
- Network address: The IP address of the machine where the 4D Server was launched.

If two servers are executed simultaneously on the same machine, the IP address must be followed by a colon and port number, for example: 192.168.92.104:19814 .

By default, the publishing port of a 4D Server is 19813. This number can be modified in the Project settings.

Once this page assigns a server, clicking the OK button will allow you to connect to the server.

If the project is published with the encryption option enabled, you must add a circumflex accent (^) before the name, otherwise the connection will be refused. For more information, refer to the Encrypting Client/Server Connections section.

Once a connection to the server has been established, the remote project will be listed on the Recent tab.

## Actualización de los archivos del proyecto en el servidor

4D Server automatically creates and sends the remote machines a [.4dz version](#) of the [.4DProject](#) project file (not compressed) in interpreted mode.

- An updated .4dz version of the project is automatically produced when necessary, *i.e.* when the project has been modified and reloaded by 4D Server. El proyecto se recarga:
  - automatically, when the 4D Server application window comes to the front of the OS or when the 4D application on the same machine saves a modification (see below).
  - when the `RELOAD PROJECT` command is executed. Calling this command is necessary for example when you have pulled a new version of the project from the source control platform.

## Actualización de los archivos de proyecto en las máquinas remotas

When an updated .4dz version of the project has been produced on 4D Server, connected remote 4D machines must log out and reconnect to 4D Server in order to benefit from the updated version.

## Utilizar 4D y 4D Server en la misma máquina

When 4D connects to a 4D Server on the same machine, the application behaves as 4D in single user mode and the design environment allows you to edit project files. This feature allows you to develop a client/server application in the same context as the deployment context.

Each time 4D performs a Save all action from the design environment (explicitly from File menu or implicitly by switching to application mode for example), 4D Server synchronously reloads project files. 4D waits for 4D Server to finish reloading the project files before it continues.

However, you need to pay attention to the following behavior differences compared to [standard project architecture](#):

- the `userPreferences.{username}` folder used by 4D is not the same folder used by 4D Server in the project folder. Instead, it is a dedicated folder, named "userPreferences", stored in the project system folder (*i.e.*, the same location as when opening a .4dz project).
- the folder used by 4D for derived data is not the folder named "DerivedData" in the project folder. Instead it is a dedicated folder named "DerivedDataRemote" located in the project system folder.
- the `catalog.4DCatalog` file is not edited by 4D but by 4D Server. Catalog information is synchronised using client/server requests
- the `directory.json` file is not edited by 4D but by 4D Server. Directory information is synchronised using client/server requests
- 4D uses its own internal components and plug-ins instead of those in 4D Server.

It is not recommended to install plug-ins or components at the 4D or 4D Server application level.

# Componentes de desarrollo

Un componente 4D es un conjunto de funciones, métodos y formularios 4D que representan una o varias funcionalidades que pueden ser [instaladas y utilizadas en aplicaciones 4D](#). Por ejemplo, puede desarrollar un componente 4D de correo electrónico que gestione todos los aspectos del envío, la recepción y el almacenamiento de correos electrónicos en aplicaciones 4D.

Puede desarrollar componentes 4D para sus propias necesidades y mantenerlos en privado. También puede [compartir sus componentes con la comunidad 4D](#).

## Definiciones

- Base proyecto: proyecto 4D utilizado para desarrollar el componente. El proyecto matriz es una base estándar sin atributos específicos. Un proyecto matricial forma un único componente.
- Proyecto local: proyecto aplicación en la que se instala y utiliza un componente.
- Componente: proyecto matricial, compilado o [generado](#), copiado en la carpeta `Components` de la aplicación local y cuyo contenido se utiliza en la aplicación local.

## Básicos

La creación e instalación de los componentes 4D se realiza directamente desde 4D:

- Para instalar un componente, basta con copiar los archivos del componente en la carpeta `Components` del proyecto. Puede utilizar alias o atajos.
- Un proyecto puede ser a la vez matriz y local, es decir, que un proyecto matriz puede utilizar a su vez uno o varios componentes. Sin embargo, un componente no puede utilizar subcomponentes por sí mismo.
- Un componente puede llamar a la mayoría de los elementos de 4D: clases, funciones, métodos proyecto, formularios proyecto, barras de menú, listas de selección, etc. No puede llamar a los métodos base ni a los triggers.
- No es posible utilizar el datastore, las tablas estándar o los archivos de datos en los componentes 4D. Sin embargo, un componente puede crear y/o utilizar tablas, campos y archivos de datos utilizando mecanismos de bases externas. Se trata de bases 4D independientes con las que se trabaja utilizando comandos SQL.
- Un proyecto local que se ejecuta en modo interpretado puede utilizar componentes interpretados o compilados. Un proyecto local que se ejecuta en modo compilado no puede utilizar componentes interpretados. En este caso, sólo se pueden utilizar componentes compilados.

## Alcance de los comandos del lenguaje

A excepción de los [comandos no utilizables](#), un componente puede utilizar cualquier comando del lenguaje 4D.

Cuando se llaman comandos desde un componente, se ejecutan en el contexto del componente, excepto el comando `EXECUTE METHOD` o `EXECUTE FORMULA` que utilizan el contexto del método especificado por el comando. También hay que tener en cuenta que los comandos de lectura del tema "Usuarios y grupos" se pueden utilizar desde un componente, pero leerán los usuarios y grupos del proyecto local (un componente no tiene sus propios usuarios y grupos).

Los comandos `SET DATABASE PARAMETER` y `Get database parameter` son una excepción: su alcance es global a la aplicación. Cuando estos comandos se llaman desde un componente, se aplican al proyecto de la aplicación local.

Además, se han especificado medidas específicas para los comandos `Structure file` y `Get 4D folder` cuando se utilizan en el marco de los componentes.

El comando `COMPONENT LIST` puede utilizarse para obtener la lista de componentes cargados por el proyecto local.

## Comandos no utilizables

Los siguientes comandos no son compatibles para su uso dentro de un componente porque modifican el archivo de

estructura - que está abierto en sólo lectura. Su ejecución en un componente genera el error -10511, "El comando CommandName no puede ser llamado desde un componente":

- ON EVENT CALL
- Method called on event
- SET PICTURE TO LIBRARY
- REMOVE PICTURE FROM LIBRARY
- SAVE LIST
- ARRAY TO LIST
- EDIT FORM
- CREATE USER FORM
- DELETE USER FORM
- CHANGE PASSWORD
- EDIT ACCESS
- Set group properties
- Set user properties
- DELETE USER
- CHANGE LICENSES
- BLOB TO USERS
- SET PLUGIN ACCESS

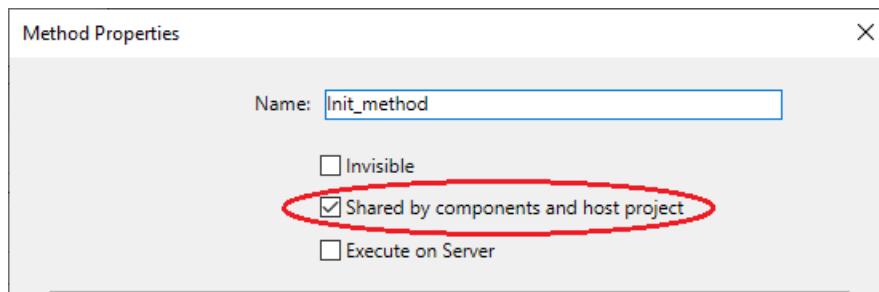
Notas:

- El comando Current form table devuelve Nil cuando se llama en el contexto de un formulario proyecto. Por consiguiente, no puede utilizarse en un componente.
- Los comandos SQL de definición de datos ( CREATE TABLE , DROP TABLE , etc.) no pueden utilizarse en el proyecto componente. Sin embargo, se soportan con bases de datos externas (ver el comando SQL CREATE DATABASE ).

## Compartir métodos proyecto

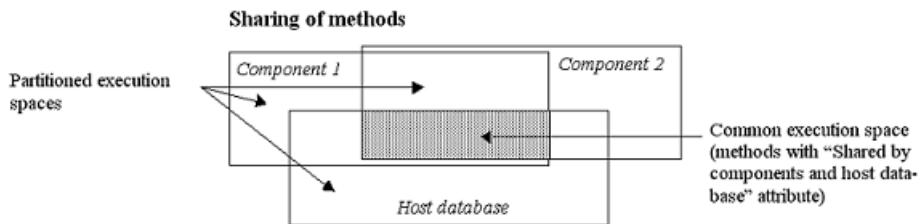
Todos los métodos proyecto de un proyecto matricial son por definición incluidos en el componente (el proyecto es el componente), lo que significa que pueden ser llamados y ejecutados dentro del componente.

Por otro lado, por defecto estos métodos proyecto no serán visibles, y no podrán ser llamados por el proyecto local. En el proyecto matriz, debe designar explícitamente los métodos que desea compartir con el proyecto local marcando la casilla Compartido por los componentes y el proyecto local en la caja de diálogo de las propiedades del método:



Los métodos proyecto compartidos se pueden llamar en el código del proyecto local (pero no se pueden modificar en el editor de métodos del proyecto local). Estos métodos son los puntos de entrada del componente.

Por el contrario, por razones de seguridad, por defecto un componente no puede ejecutar métodos proyecto que pertenezcan al proyecto local. En algunos casos, puede ser necesario permitir que un componente acceda a los métodos proyecto de su proyecto local. Para ello, debe designar explícitamente qué métodos proyecto del proyecto local quiere hacer accesibles a los componentes (en las propiedades del método, marque la casilla Compartido por componentes y proyecto local).



Una vez que los métodos del proyecto anfitrión están disponibles para los componentes, se puede ejecutar un método anfitrión desde dentro de un componente utilizando los comandos `EXECUTE FORMULA` o `EXECUTE METHOD`. Por ejemplo:

```
// Método local
component_method("host_method_name")
```

```
// component_method
C_TEXT($1)
EXECUTE METHOD($1)
```

Una base local interpretada que contenga componentes interpretados puede ser compilada o verificada sintácticamente si no llama a métodos del componente interpretado. En caso contrario, aparecerá una caja de diálogo de advertencia al intentar lanzar la compilación o una comprobación de sintaxis y no será posible realizar la operación.

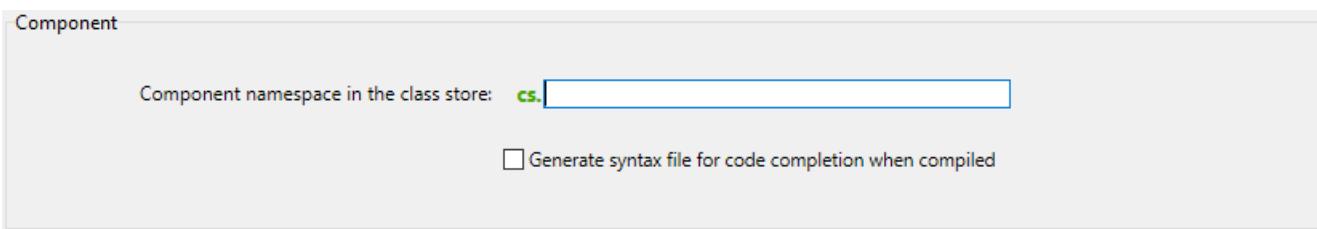
Tenga en cuenta que un método interpretado puede llamar a un método compilado, pero no a la inversa, salvo mediante el uso de los comandos `EXECUTE METHOD` y `EXECUTE FORMULA`.

## Compartir las clases y las funciones

Por defecto, las clases y funciones de los componentes no pueden ser llamadas desde el editor de métodos 4D del proyecto local. Si quiere exponer las clases y funciones del componente en el proyecto local, necesita declarar un espacio de nombres del componente. Además, puede controlar cómo se sugieren las clases y las funciones de los componentes en el editor de métodos local.

### Declarar del namespace

Para permitir que las clases y las funciones de su componente se expongan en los proyectos locales, introduzca un valor en la opción **Componente namespace en la class store** en la página **General** de las Propiedades del proyecto utilizado como matriz. Por defecto, el área está vacía: las clases de componentes no están disponibles fuera del contexto de los componentes.



Un *namespace* garantiza que no surja ningún conflicto cuando un proyecto local utilice diferentes componentes que tienen clases o funciones con nombres idénticos. Un namespace del componente debe ser compatible con [reglas de denominación de las propiedades](#).

Cuando introduce un valor, declare que las clases y las funciones del componente estarán disponibles en la [user class store \(cs\)](#) del código del proyecto local, a través del namespace `cs.<value>`. Por ejemplo, si introduce "eGeometry" como namespace del componente, asumiendo que ha creado una clase `Rectangle` que contiene una función `getArea()`, una vez que su proyecto se instala como componente, el desarrollador del proyecto local puede escribir:

```
//en el proyecto local
var $rect: cs.eGeometry.Rectangle
$rect:=cs.eGeometry.Rectangle.new(10;20)
$area:=$rect.getArea()
```

Por supuesto, se recomienda utilizar un nombre distintivo para evitar cualquier conflicto. Si en el proyecto ya existe una clase usuario con el mismo nombre que un componente, se tiene en cuenta la clase usuario y se ignoran las clases del componente.

Las clases ORDA de un componente no están disponibles en el proyecto local. Por ejemplo, si hay una dataclass llamada Employees en su componente, no podrá utilizar una clase "cs.Mycomponent.Employee" en el proyecto local.

## Clases ocultas

Como en todo proyecto, puede crear clases y funciones ocultas en el componente anteponiendo a los nombres un guión bajo ("\_"). Cuando se define un [namespace de componente](#), las clases y funciones ocultas del componente no aparecerán como sugerencias al utilizar completar el código.

Sin embargo, hay que tener en cuenta que pueden seguir utilizándose si se conocen sus nombres. Por ejemplo, la siguiente sintaxis es válida incluso si la clase `_Rectangle` está oculta:

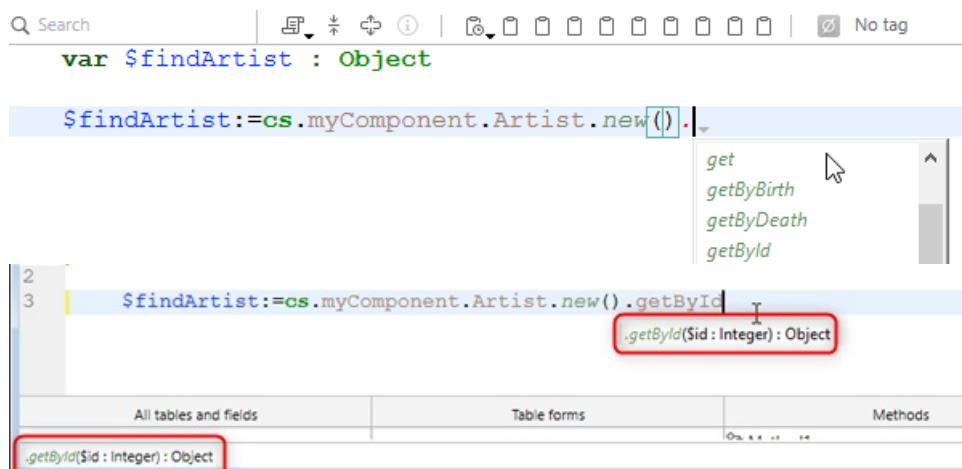
```
$rect:=cs.eGeometry._Rectangle.new(10;20)
```

Las funciones no ocultas al interior de una clase oculta aparecen como sugerencias cuando se utiliza completar código con una clase que [hereda](#) de ella. Por ejemplo, si un componente tiene una clase `Teacher` que hereda una clase `_Person`, la finalización del código para `Teacher` sugiere funciones no ocultas de `_Person`.

## Completar el código de los componentes compilados

Para facilitar el uso de su componente a los desarrolladores, puede marcar la opción [Generate syntax file for code completion when compiled](#) en la página [General](#) de las Propiedades del proyecto utilizadao como matriz.

Un archivo de sintaxis (formato JSON) se crea automáticamente durante la fase de compilación, llena con la sintaxis de las clases, funciones y [métodos exposed](#) de su componente y colocados en la carpeta `¥Resources¥en.lproj` del proyecto del componente. 4D utiliza el contenido de ese archivo de sintaxis para generar ayuda contextual en el editor de código, como la finalización del código y la sintaxis de las funciones:



Si no ingresa un [namespace](#), los recursos de las clases y de los métodos exposed no se generan incluso si la opción de archivo de sintaxis está marcada.

## Paso de variables

Las variables locales, proceso e interproceso no se comparten entre los componentes y los proyectos locales. La única forma de modificar las variables del componente desde el proyecto local y viceversa es utilizando punteros.

Ejemplo utilizando un array:

```
//En el proyecto local: ARRAY INTEGER( MyArray;10)
AMethod(> MyArray)

//En el componente, el método proyecto AMethod contiene:
APPEND TO ARRAY($1->;2)
```

Ejemplos utilizando variables:

```
C_TEXT(myvariable)
component_method1(>myvariable)
```

```
C_POINTER($p)
$p:=component_method2(...)
```

Sin un puntero, un componente puede seguir accediendo al valor de una variable de la base local (pero no a la propia variable) y viceversa:

```
//En la base local
C_TEXT($input_t)
$input_t:="DoSomething"
component_method($input_t)
// component_method obtiene "DoSomething" en $1 (pero no la variable $input_t)
```

Cuando se utilizan punteros para que los componentes y el proyecto local se comuniquen, hay que tener en cuenta las siguientes particularidades:

- El comando `Get pointer` no devolverá un puntero a una variable del proyecto local si se llama desde un componente y viceversa.
- La arquitectura de componentes permite la coexistencia, dentro del mismo proyecto interpretado, de componentes interpretados y compilados (a la inversa, en un proyecto compilado sólo pueden utilizarse componentes compilados). Para utilizar punteros en este caso, debe respetar el siguiente principio: el intérprete puede desanclar un puntero construido en modo compilado; sin embargo, en modo compilado, no puede desanclar un puntero construido en modo interpretado. Ilustraremos este principio con el siguiente ejemplo: dados dos componentes, C (compilado) e I (interpretado), instalados en el mismo proyecto local.
- Si el componente C define la variable `myCvar`, el componente I puede acceder al valor de esta variable utilizando el puntero `>myCvar`.
- Si el componente C define la variable `myIvar`, el componente C no puede acceder a esta variable utilizando el puntero `>myIvar`. Esta sintaxis provoca un error de ejecución.
- La comparación de punteros utilizando el comando `RESOLVE POINTER` no se recomienda con los componentes, ya que el principio de partición de variables permite la coexistencia de variables con el mismo nombre pero con contenidos radicalmente diferentes en un componente y en el proyecto local (u otro componente). El tipo de la variable puede incluso ser diferente en ambos contextos. Si los punteros `myptr1` y `myptr2` apuntan cada uno a una variable, la siguiente comparación producirá un resultado incorrecto:

```
RESOLVE POINTER(myptr1;vVarName1;vtablenum1;vfieldnum1)
RESOLVE POINTER(myptr2;vVarName2;vtablenum2;vfieldnum2)
If(vVarName1=vVarName2)
//Esta prueba devuelve True aunque las variables sean diferentes
```

En este caso, es necesario utilizar la comparación de punteros:

```
If(myPtr1==myPtr2) //Esta prueba devuelve False
```

## Gestión de errores

Un [método de gestión de errores](#) instalado por el comando `ON ERR CALL` sólo se aplica a la aplicación en ejecución. En el caso de un error generado por un componente, no se llama al método de gestión de errores `ON ERR CALL` del proyecto local, y viceversa.

## Acceso a las tablas del proyecto local

Aunque los componentes no pueden utilizar tablas, los punteros pueden permitir que los proyectos locales y los componentes se comuniquen entre sí. Por ejemplo, este es un método que podría ser llamado desde un componente:

```
// llamar a un método componente  
methCreateRec(→[PEOPLE];→[PEOPLE]Name;"Julie Andrews")
```

Dentro del componente, el código del método `methCreateRec` :

```
C_POINTER($1) //Puntero a una tabla del proyecto local  
C_POINTER($2) //Puntero a un campo del proyecto local  
C_TEXT($3) // Valor a insertar  
  
$tablepointer:=$1  
$fieldpointer:=$2  
CREATE RECORD($tablepointer->)  
  
$fieldpointer->:=$3  
SAVE RECORD($tablepointer->)
```

En el contexto de un componente, 4D asume que una referencia a un formulario tabla es una referencia al formulario tabla local (ya que los componentes no pueden tener tablas.)

## Uso de tablas y campos

Un componente no puede utilizar las tablas y campos definidos en la estructura 4D del proyecto matriz. Sin embargo, puede crear y utilizar bases externas, y luego utilizar sus tablas y campos según sus necesidades. Puede crear y gestionar bases externas utilizando SQL. Sin embargo, puede crear y utilizar bases externas, y luego utilizar sus tablas y campos según sus necesidades. Utilizar una base externa significa designar temporalmente esta base como base actual, es decir, como la base de destino para las consultas SQL ejecutadas por 4D. Las bases externas se crean con el comando SQL `CREATE DATABASE`.

### Ejemplo

El siguiente código se incluye en un componente y realiza tres acciones básicas con una base de datos externa:

- creación de la base de datos externa si aún no existe,
- añade datos a la base de datos externa,
- lectura de datos desde la base de datos externa.

Creación de la base de datos externa:

```

<>MyDatabase:=Get 4D folder+"\MyDB" // (Windows) almacena los datos en un directorio autorizado
Begin SQL
    CREATE DATABASE IF NOT EXISTS DATAFILE :[<>MyDatabase];
    USE DATABASE DATAFILE :[<>MyDatabase];
    CREATE TABLE IF NOT EXISTS KEEPIT
    (
        ID INT32 PRIMARY KEY,
        kind VARCHAR,
        name VARCHAR,
        code TEXT,
        sort_order INT32
    );

    CREATE UNIQUE INDEX id_index ON KEEPIT (ID);

    USE DATABASE SQL_INTERNAL;

End SQL

```

Escritura en la base de datos externa:

```

$Ptr_1:=$2 // recupera datos del proyecto local mediante punteros
$Ptr_2:=$3
$Ptr_3:=$4
$Ptr_4:=$5
$Ptr_5:=$6
Begin SQL

    USE DATABASE DATAFILE :[<>MyDatabase];

    INSERT INTO KEEPIT
    (ID, kind, name, code, sort_order)
    VALUES
    (:[$Ptr_1], :[$Ptr_2], :[$Ptr_3], :[$Ptr_4], :[$Ptr_5]);

    USE DATABASE SQL_INTERNAL;

End SQL

```

Lectura en una base de datos externa:

```

$Ptr_1:=$2 // accede a los datos del proyecto local a través de punteros
$Ptr_2:=$3
$Ptr_3:=$4
$Ptr_4:=$5
$Ptr_5:=$6

Begin SQL

    USE DATABASE DATAFILE :[<>MyDatabase];

    SELECT ALL ID, kind, name, code, sort_order
    FROM KEEPIT
    INTO :$Ptr_1, :$Ptr_2, :$Ptr_3, :$Ptr_4, :$Ptr_5;

    USE DATABASE SQL_INTERNAL;

End SQL

```

## Utilización de formularios

- Sólo los "formularios de proyecto" (formularios que no están asociados a ninguna tabla específica) pueden utilizarse en un componente. Sólo los "formularios de proyecto" (formularios que no están asociados a ninguna tabla específica) pueden utilizarse en un componente.
- Un componente puede llamar a formularios tabla del proyecto local. Tenga en cuenta que en este caso es necesario utilizar punteros en lugar de nombres de tablas entre paréntesis [] para especificar los formularios en el código del componente.

Si un componente utiliza el comando `ADD RECORD`, se mostrará el formulario de entrada actual del proyecto local, en el contexto del proyecto local. Por consiguiente, si el formulario incluye variables, el componente no tendrá acceso a ellas.

- Puede publicar formularios de componentes como subformularios en los proyectos locales. Esto significa que puede, más concretamente, desarrollar componentes que ofrezcan objetos gráficos. Por ejemplo, los Widgets que ofrece 4D se basan en el uso de subformularios en los componentes.

En el contexto de un componente, cualquier formulario de proyecto referenciado debe pertenecer al componente. Por ejemplo, dentro de un componente, hacer referencia a un formulario proyecto local utilizando `DIALOG` u `Open form window` arrojará un error.

## Utilización de recursos

Los componentes pueden utilizar recursos situados en la carpeta Resources del componente.

Los mecanismos automáticos son operacionales: los archivos XLIFF encontrados en la carpeta Resources de un componente serán cargados por este componente.

En un proyecto local que contiene uno o más componentes, cada componente, así como los proyectos locales, tiene su propia "cadena de recursos." Los recursos están divididos entre las diferentes proyectos: no es posible acceder a los recursos del componente A desde el componente B o desde el proyecto local.

## Ejecución del código de inicialización

Un componente puede ejecutar automáticamente código 4D al abrir o cerrar la base local, por ejemplo para cargar y/o guardar las preferencias o los estados usuario relacionados con el funcionamiento de la base local.

La ejecución del código de inicialización o cierre se realiza mediante el método base `On Host Database Event`.

Por razones de seguridad, debe autorizar explícitamente la ejecución del método base `On Host Database Event` en la base local para poder llamarlo. Por razones de seguridad, debe autorizar explícitamente la ejecución del método base `On Host Database Event` en la base local para poder llamarlo.

## Protección de los componentes: compilación

Por defecto, todo el código de un proyecto matriz instalado como componente es potencialmente visible desde el proyecto local. En particular:

- Los métodos proyecto compartido se encuentran en la Página Métodos del Explorador y pueden ser llamados en los métodos del proyecto local. Su contenido puede ser seleccionado y copiado en el área de vista previa del Explorador. También se pueden ver en el depurador. Sin embargo, no es posible abrirlos en el editor de métodos o modificarlos.
- Los otros métodos proyecto del proyecto matriz no aparecen en el Explorador, pero también pueden verse en el depurador del proyecto local.
- Las clases y funciones no ocultas pueden verse en el depurador [si se declara un namespace](#).

Para proteger eficazmente el código de un componente, basta con [compilar y generar](#) el proyecto utilizado como matriz y proporcionarlo en forma de archivo .4dz. Cuando se instala un proyecto matricial compilado como un componente:

- Los métodos, clases y funciones del proyecto compartido pueden ser llamados en los métodos del proyecto local. Los métodos de proyecto compartidos también son visibles en la página de Métodos del explorador. Sin embargo, su contenido no aparecerá en el área de vista previa ni en el depurador.
- Los otros métodos del proyecto matriz nunca aparecerán.

## Compartir componentes

Lo animamos a que apoye a la comunidad de desarrolladores 4D compartiendo sus componentes, preferiblemente en la plataforma [GitHub](#). Recomendamos que utilice el tema `4d-component` para ser referenciado correctamente.

# Plugins de desarrollo

## ¿Por qué es necesario un plug-in?

Aunque 4D ofrece cientos de métodos integrados para manipular objetos, registros e implementar la interfaz de usuario, es posible que se necesite algún uso o característica especial (que a veces depende de la plataforma): uno puede necesitar ODBC en Windows, otro puede necesitar los servicios de Apple en macOS, mientras que otro puede querer implementar herramientas estadísticas específicas, inicio de sesión en redes sociales, plataforma de pago, acceso a archivos a través de la red, una interfaz de usuario especial o una estructura de imagen privada.

Es evidente que cubrir todas las áreas de los sistemas operativos macOS y Windows por medio de los comandos de 4D sin duda conduciría a un producto con miles de comandos, y al mismo tiempo, la mayoría de los usuarios no tendrían necesidad de un conjunto tan grande de funcionalidades. Además, la creación de una herramienta tan completa haría que el entorno 4D fuera increíblemente complejo y llevaría a la mayoría de los usuarios meses de estudio antes de poder esperar resultados útiles.

La naturaleza modular del entorno 4D permite la creación de aplicaciones básicas, pero no impide el desarrollo de sistemas muy complejos. La arquitectura del plug-in 4D abre el entorno 4D a todo tipo de aplicaciones o de usuario. Los plug-ins 4D multiplican la potencia y la productividad de la aplicación o del usuario.

## ¿Qué es un plug-in y qué puede hacer?

Un plug-in es una pieza de código que 4D lanza al inicio. Añade funcionalidad a 4D y aumenta así su capacidad.

Normalmente, un plug-in hace cosas que:

- 4D no puede efectuar (es decir, una tecnología de plataforma específica),
- será muy difícil de escribir sólo con 4D,
- sólo están disponibles como punto de entrada del plug-in

Un plug-in suele contener un conjunto de rutinas entregadas al desarrollador 4D. Puede manejar un Área Externa y ejecutar un proceso externo.

- Una rutina de conexión es una rutina escrita en lenguaje nativo (normalmente C o C++) que provoca una acción.
- Un área externa es una parte de un formulario que puede mostrar casi todo e interactuar con el usuario cuando sea necesario.
- Un proceso externo es un proceso que se ejecuta solo, normalmente en un bucle, haciendo casi todo lo que quiere. Todo el código del proceso pertenece al plug-in, 4D simplemente está presente para recibir/enviar eventos al proceso.

## Nota importante

Un plug-in puede ser muy sencillo, con una sola rutina que realice una tarea muy pequeña, o puede ser muy complejo, con cientos de rutinas y áreas. Prácticamente no hay límite para lo que puede hacer un plug-in, sin embargo todo desarrollador de plug-ins debe recordar que un plug-in es una parte de código "de muestra". Es el plug-in que se ejecuta dentro de 4D, no lo contrario. Como parte de código, es el anfitrión de 4D; no es una aplicación independiente. Comparte el tiempo de la CPU y la memoria con 4D y otros plug-ins, por lo tanto, debería ser un código conciso, utilizando sólo lo necesario para funcionar. Por ejemplo, en los bucles largos, un plug-in debe llamar a `PA_Yield()` para dar tiempo al planificador 4D a menos que su tarea sea crítica tanto para él como para la aplicación.

## ¿Cómo crear un plug-in?

4D ofrece en GitHub un código abierto [plug-in SDK](#), que contiene el plug-in API 4D y el asistente de plugins 4D:

- el [\\*\\*Plugin API de 4D \\*\\*](#), escrito en C, añade más de 400 funciones que le ayudan a crear fácilmente sus propios plug-ins para añadir nuevas funcionalidades a su aplicación 4D. Las funciones del plug-in de API de 4D gestionan

todas las interacciones entre la aplicación 4D y su plug-in.

- [El Asistente de plug-in 4D](#) es una herramienta esencial que simplifica la tarea de desarrollar plugins 4D. Escribe el código que 4D necesita para cargar e interactuar correctamente con un plug-in, permitiéndole concentrarse en su propio código.

## Compartir los plug-ins

We encourage you to support the 4D developer community by sharing your plug-ins, preferably on the [GitHub platform](#). We recommend that you use the `4d-plugin` topic to be correctly referenced.

# Comencemos

4D View Pro is a [4D component](#) that includes a [4D form area](#) and specific [methods](#). It allows you to embed advanced spreadsheet features in your projects.

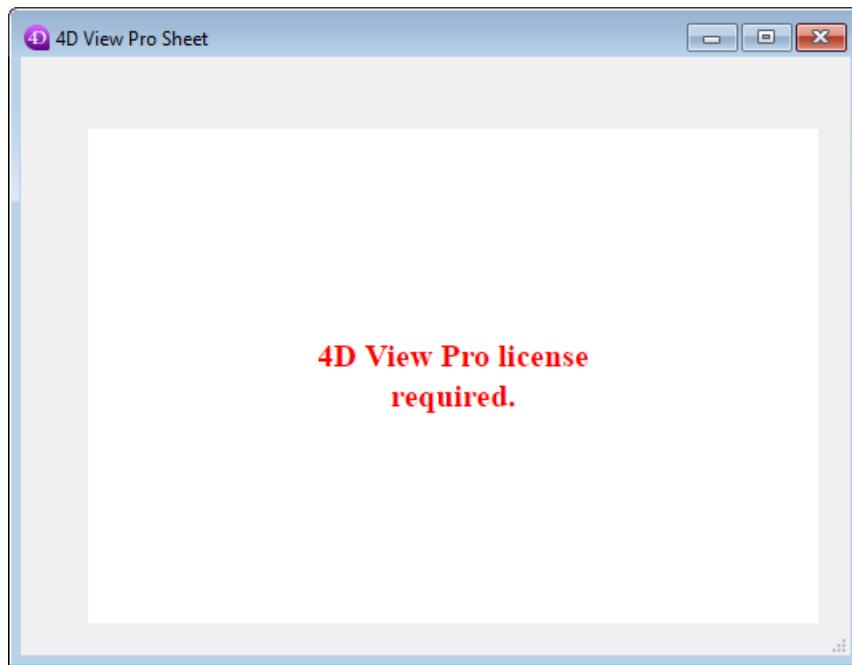
Una hoja de cálculo es una aplicación que contiene una cuadrícula de celdas en las que se puede introducir información, ejecutar cálculos o mostrar imágenes. 4D View Pro is powered by the [SpreadJS spreadsheet solution](#) integrated in 4D.

Embedding 4D View Pro areas in your forms allows you to import and export spreadsheets documents using the 4D View Pro commands.

## Instalación y activación

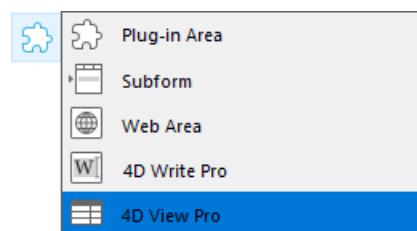
4D View Pro features are directly included in 4D, making it easy to deploy and manage. No se requiere ninguna instalación adicional.

Sin embargo, 4D View Pro requiere una licencia. You need to activate this license in your application in order to use its features. When using this component without a license, the contents of an object that requires a 4D View Pro feature are not displayed at runtime, an error message is displayed instead:



## Inserción de un área 4D View Pro

4D View Pro documents are displayed and edited manually in a [4D form object](#) named 4D View Pro. To select this object, click on the last tool in the object bar:



You can also select a preconfigured 4D View Pro area in the [Object library](#).

4D View Pro areas can also be [created and used offscreen](#).

You can [configure the area](#) using the Property List and 4D View Pro methods.

## Fundamentos de la selección, la entrada y de la navegación

Las hojas de cálculo se componen de líneas y columnas. A cada línea se le asocia un número. A letter (or group of letters once the number of columns surpasses the number of letters in the alphabet) is associated with each column. The intersection of a row and a column makes a cell. Las celdas pueden ser seleccionadas y sus contenidos editados.

### Selecting cells, columns and rows

- To select a cell, simply click on it or use the direction arrows on the keyboard. Its content (or formula) is displayed within the cell.
- To select several continuous cells, drag the mouse from one end of the selection to the other. You can also click on the two ends of the selection while holding down the Shift key.
- To select all cells in the spreadsheet, click on the cell at the top left of the area:



- To select a column, click on the corresponding letter (or set of letters).
- To select a row, click on the corresponding number.
- To select a group of cells that are not continuous, hold down the Ctrl key (Windows) or Command key (Mac) and click on each cell to be selected.
- To deselect cells, simply click anywhere within the spreadsheet.

### Entrada de datos

Double-clicking on a cell allows passing into input mode in the relevant cell. If the cell is not empty, the insertion cursor is placed after the content of the cell.



Data can be entered directly once a cell is already selected, even if the insertion cursor is not visible. The input then replaces the content of the cell.

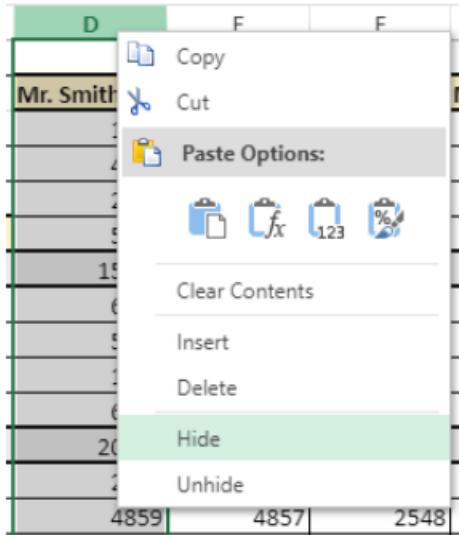
The Tab key validates the cell input and selects the cell to its right. Combining the Shift + Tab keys validates the cell input and selects the cell to its left.

The Carriage return key validates the cell input and selects the cell below it. Combining the Shift + Carriage return keys validates the cell input and selects the cell above it.

The direction keys (arrows) allow you to move a cell in the direction indicated by the arrow.

### Utilización del menú contextual

4D View Pro areas benefit from an automatic context menu that offers standard editing features such as copy and paste, but also basic spreadsheet features:



The Copy/Cut and Paste features of the context menu only work within the spreadsheet area, they do not have access to the system pasteboard. System shortcuts such as Ctrl+c/Ctrl+v works however and can be used to exchange data between the area and other applications.

Depending on the clicked area, the following options are also available:

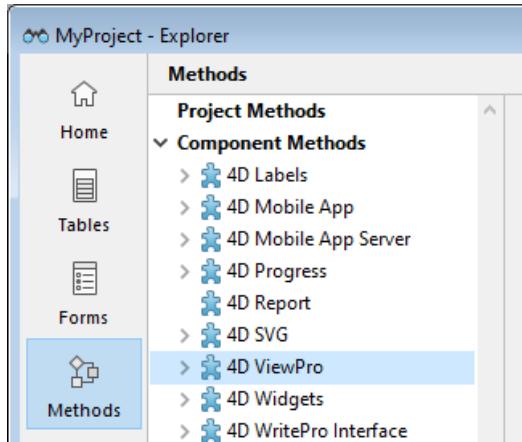
- click on a column or row header: Insert, Delete, Hide, or Unhide the contents
- haga clic en una celda o en un rango de celdas:
  - Filter: allows hiding row through filters (see [Filtering rows](#) in the SpreadJS documentation).
  - Sort: sorts the column contents.
  - Insert Comment: allows user to enter a comment for an area. When a comment has been entered for an area, the top left cell of the area displays a small red triangle:

8355	7195
------	------

## Utilización de los métodos 4D View Pro

4D View Pro methods can be used in the 4D Method editor, just like 4D language commands.

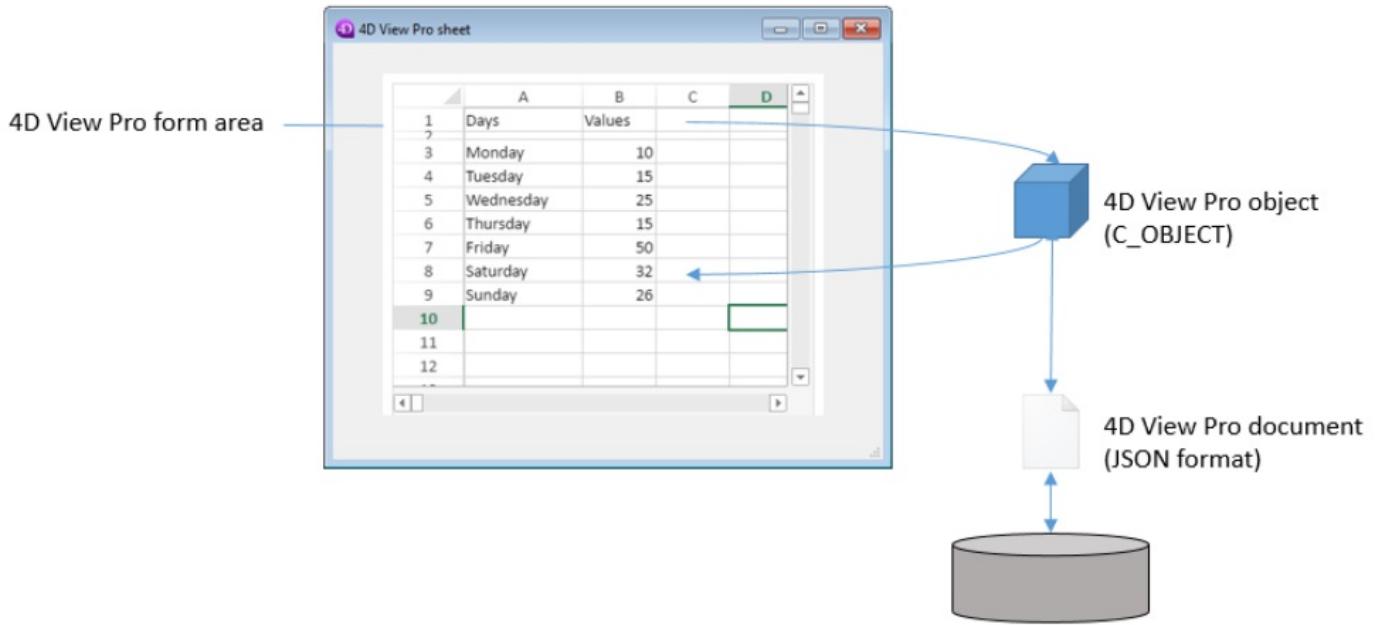
Since 4D View Pro is a built-in 4D component, you can access its list of methods from the Explorer, in the Component Methods section:



For a detailed list of component methods, see [Method list](#).

## Abordando un área 4D View Pro

A 4D View Pro area handles several objects and elements.



Most of 4D View Pro methods require a `vpAreaName` parameter, which is the [4D View Pro form area name](#) (4D form object). This name is the [object name](#) property.

For example, if you want to set the total number of columns of an area named "myVpArea", you write:

```
VP SET COLUMN COUNT("myVpArea";5)
```

When loading a 4D View Pro object in a form area, 4D generates the [On VP Ready](#) form event once the whole area is loaded. You must execute any 4D View Pro code handling the area in this event, otherwise an error is returned.

## Utilización de objetos de rango

Some 4D View Pro methods require a `rangeObj` parameter. In 4D View Pro, a range is an object that references an area in a spreadsheet. Esta área puede estar compuesta de una o varias celdas. Using 4D View Pro methods, you can create ranges and pass them to other methods to read from or write to specific locations in your document.

For example, to create a range object for the following cells:

You can use the [VP Cells](#) method:

```
var $myRange : Object
$myRange:=VP Cells("ViewProArea";2;4;2;3) // C5 a D7
```

You can then pass `$myRange` to another 4D View Pro method to modify these cells (for example add a border to the set of cells with [VP SET BORDER](#)).

4D View Pro range objects are composed of several properties:

- area - El nombre del área 4D View Pro
- rangos - Una colección de objeto(s) rango. Available properties within each range object depend on the range object type. For example, a column range object will only include the *.column* and *.sheet* properties.

Propiedad		Tipo	Descripción	Disponible para
area		texto	Nombre de objeto formulario área 4D View Pro	always available
ranges		colección	Colección de rangos	always available
	[ ].name	texto	Nombre de rango	name
	[ ].sheet	number	Sheet index (current sheet index by default) (counting begins at 0)	celda, celdas, línea, líneas, columna, columnas, todos, nombre
	[ ].row	number	Índice de la línea (el conteo comienza en 0)	celda, celdas, línea, líneas
	[ ].rowCount	number	Número de líneas	celdas, líneas
	[ ].column	number	Índice de la columna (el conteo comienza en 0)	celda, celdas, columna, columnas
	[ ].columnCount	number	Conteo de columnas	celdas, columnas

## Importar y exportar documentos

4D View Pro supports the import and export of several document formats:

- .4vp
- .xlsx
- .txt y .csv
- .pdf (sólo para exportación)

For more details, check out the description of [VP IMPORT DOCUMENT](#) and [VP EXPORT DOCUMENT](#).

# Configuring 4D View Pro Areas

The 4D View Pro area properties can be configured using the Property list. Spreadsheet properties are available through the language.

## Propiedades del área de formulario

Using the area's property list, you can set [4D View Pro object properties](#) such as Object Name, [Variable or Expression](#), Appearance, Action, and Events.

### Selección de una interfaz usuario

You can select the interface to use with your 4D View Pro form areas in the [Property List](#), under Appearance:



You can also use the `userInterface` and `withFormulaBar` (only with the "toolbar" interface) JSON properties.

Interfaces allow for basic modifications and data manipulation. User-defined modifications are saved in the 4D View Pro object when the user saves the document.

### Cinta

### Barra de herramientas

Enabling the Toolbar interface displays the [Show Formula Bar](#) option. When selected, the formula bar is visible below the Toolbar interface.

Con barra de fórmula visible:

### Funcionalidades

Both the Ribbon and the Toolbar interfaces group related features into tabs:

Pestaña	Acciones	Interfaz Cinta	Interfaz Barra de herramientas
File	Gestión de archivos	X	
Inicio	Apariencia del texto	X	X
Insertar	Añadir elementos	X	X
Fórmulas	Formula calculation and library	X	X
Datos	Gestión de los datos	X	X
Mostrar	Presentación visual	X	X
Settings	Presentación de la hoja	X	

## Eventos formulario

The following form events are available in the Property List for 4D View Pro areas.

Some of the events are standard form events (available to all active objects) and some are specific 4D View Pro form events. Some standard form events provide extended information in the object returned by the [FORM Event](#) command when they are generated for 4D View Pro areas. The following table shows which events are standard and which are specific or provide additional information to 4D View Pro areas:

Evento 4D estándar	Eventos 4D View Pro específicos y extendidos
On Load	<a href="#">On VP Ready</a>
On Getting Focus	<a href="#">On Clicked</a>
On Losing Focus	<a href="#">On Double Clicked</a>
On Unload	<a href="#">On Header Click</a>
	<a href="#">On After Edit</a>
	<a href="#">On Selection Change</a>
	<a href="#">On Column Resize</a>
	<a href="#">On Row Resize</a>
	<a href="#">On VP Range Changed</a>

## Opciones hoja

The 4D View Pro sheet options object allows you to control various options of your 4D View Pro areas. This object is handled by the following commands:

- [VP SET SHEET OPTIONS](#)
- [VP Get sheet options](#)

## Apariencia de la hoja

Propiedad		Tipo	Descripción
allowCellOverflow		booleano	Specifies whether data can overflow into adjacent empty cells.
sheetTabColor		cadena	A color string used to represent the sheet tab color, such as "red", "#FFFF00", "rgb(255,0,0)", "Accent 5", and so on.
frozenlineColor		cadena	A color string used to represent the frozen line color, such as "red", "#FFFF00", "rgb(255,0,0)", "Accent 5", and so on.
clipBoardOptions		entero largo	La opción portapapeles. Valores disponibles: <code>vk clipboard paste options all</code> , <code>vk clipboard paste options formatting</code> , <code>vk clipboard paste options formulas</code> , <code>vk clipboard paste options formulas and formatting</code> , <code>vk clipboard paste options values</code> , <code>vk clipboard paste options values and formatting</code>
rejilla		objeto	Las opciones de la línea de rejilla.
	color	cadena	A color string used to represent the grid line color, such as "red", "#FFFF00", "rgb(255,0,0)", "Accent 5", and so on.
	showVerticalGridline	booleano	Specifies whether to show the vertical grid line.
	showHorizontalGridline	booleano	Specifies whether to show the horizontal grid line.
rowHeaderVisible		booleano	Especifica si el encabezado de la línea es visible.
colHeaderVisible		booleano	Especifica si el encabezado de la columna es visible.
rowHeaderAutoText		entero largo	Specifies whether the row header displays letters or numbers or is blank. Valores disponibles: <code>vk header auto text blank</code> , <code>vk header auto text letters</code> , <code>vk header auto text numbers</code>
colHeaderAutoText		entero largo	Specifies whether the column header displays letters or numbers or is blank. Valores disponibles: <code>vk header auto text blank</code> , <code>vk header auto text letters</code> , <code>vk header auto text numbers</code>
selectionBackColor		cadena	El color de fondo de la selección para la hoja. (formato RGBA preferido)
selectionBorderColor		cadena	El color del borde de la selección para la hoja.
sheetAreaOffset		objeto	Las opciones de sheetAreaOffset.
	left	entero largo	El desplazamiento a la izquierda de la hoja desde la local.
	top	entero largo	El desplazamiento superior de la hoja desde el local.

Todas las propiedades son opcionales.

## Protección de la hoja

To lock the whole sheet, you only need to set the `isProtected` property to true. You can then unlock cells individually by setting the `locked` cell style property.

Propiedad		Tipo	Descripción
isProtected		booleano	Specifies whether cells on this sheet that are marked as protected cannot be edited.
protectionOptions		objeto	A value that indicates the elements that you want users to be able to change. If null : the protectionOptions parameter is reset.
	allowSelectLockedCells	booleano	Specifies whether the user can select locked cells, optional. True por defecto.
	allowSelectUnlockedCells	booleano	Specifies whether the user can select unlocked cells, optional. True por defecto.
	allowSort	booleano	Specifies whether the user can sort ranges, optional. Falso por defecto.
	allowFilter	booleano	Specifies whether the user can filter ranges, optional. Falso por defecto.
	allowEditObjects	booleano	Specifies whether the user can edit floating objects, optional. Falso por defecto.
	allowResizeRows	booleano	Specifies whether the user can resize rows, optional. Falso por defecto.
	allowResizeColumns	booleano	Specifies whether the user can resize columns, optional. Falso por defecto.
	allowDragInsertRows	booleano	Specifies whether the user can perform the drag operation to insert rows, optional. Falso por defecto.
	allowDragInsertColumns	booleano	Specifies whether the user can perform the drag operation to insert columns, optional. Falso por defecto.
	allowInsertRows	booleano	Specifies whether the user can insert rows, optional. Falso por defecto.
	allowInsertColumns	booleano	Specifies whether the user can insert columns, optional. Falso por defecto.
	allowDeleteRows	booleano	Specifies whether the user can delete rows, optional. Falso por defecto.
	allowDeleteColumns	booleano	Specifies whether the user can delete columns, optional. Falso por defecto.

Todas las propiedades son opcionales.

## Formato de las celdas

Defining a format pattern ensures that the content of your 4D View Pro documents is displayed the way you intended. Formats can be set using the selected 4D View Pro [interface](#), or using the [VP SET VALUE](#) or [VP SET NUM VALUE](#) methods.

4D View Pro has built-in formats for numbers, dates, times, and text, but you can also create your own patterns to format the contents of cells using special characters and codes.

For example, when using the [VP SET VALUE](#) or [VP SET NUM VALUE](#) methods to enter amounts in an invoice, you may want the currency symbols (\$, €, ¥, etc.) to be aligned regardless of the space required by the number (i.e., whether the amount is \$5.00 or \$5,000.00). You could use formatting characters and specify the pattern (\$\* #,##0.00) which would display amounts as shown:

\$	4,180.00
\$	15.00
\$	200.00
\$	15,600.00
\$	1,672.00

Note that when creating your own format patterns, only the display of the data is modified. El valor de los datos permanece sin cambios.

## Formatos numérico y texto

Number formats apply to all number types (e.g., positive, negative, and zeros).

Carácter	Descripción	Ejemplo
0	Marcador de posición que muestra ceros.	#.00 will display 1.1 as 1.10
.	Muestra un punto decimal	0.00 will display 1999 as 1999.00
,	Muestra el separador de miles en un número. Thousands are separated by commas if the format contains a comma enclosed by number signs "#" or by zeros. A comma following a digit placeholder scales the number by 1,000.	#,0 mostrará 12200000 como 12,200,000
_	Salta el ancho del siguiente carácter.	Usually used in combination with parentheses to add left and right indents, _( and _) respectively.
@	Formatter for text. Aplica el formato a todo el texto de la celda	"[Red]@" applies the red font color for text values.
*	Repite el carácter siguiente para llenar la anchura de la columna.	0*- will include enough dashes after a number to fill the cell, whereas *0 before any format will include leading zeros.
" "	Displays the text within the quotes without interpreting it.	"8%" será mostrado como: 8%
%	Muestra los números como un porcentaje de 100.	El 8% se mostrará como 0,08
#	Dígito placeholder que no muestra ceros adicionales. Si un número tiene más dígitos a la derecha del decimal que los placeholders, el número es redondeado.	#.# mostrará 1.54 como 1.5
?	Dígito placeholder que留下 espacio para ceros adicionales, pero no los muestra. Normalmente se utiliza para alinear números por punto decimal.	\$?? muestra un máximo de 2 decimales y hace que los signos de dólar se alineen para cantidades variables.
¥	Muestra el carácter que sigue.	#.00? #.00? #.00? #.00? #.00? will display 123 as 123.00?
/	Cuando se usa con números, los muestra como fracciones. Cuando se usa con texto, fechas o códigos de tiempo, se muestra "as-is".	#/# mostrará .75 como 3/4
[ ]	Crea formatos condicionales.	[>100][GREEN]#,##0;[<=100][YELLOW]#,##0;[BLUE]#,##0
E	Formato notación científica.	#E+# - mostrará 2E+6 en lugar de 1,500,500
[color]	Formatea el texto o el número en el color especificado	[Green]###.###[Red]-###.###[Blue]

Ejemplo

```
//Set the cell value as $125,571.35
VP SET VALUE(VP Cell("ViewProArea";3;2);New object("value";125571.35;"format";"$_(* #,##0.00_)")
```

## Formatos fecha y hora

4D View Pro provides the following constants for ISO 8601 date and time patterns:

Constante	Valor	Comentario
<code>vk pattern full date time</code>	<code>"fullDateTimePattern"</code>	ISO 8601 format for the full date and time in current localization. Patrón por defecto USA: "dddd, dd MMMM yyyy HH:mm:ss"
<code>vk pattern long date</code>	<code>"longDatePattern"</code>	ISO 8601 format for the full date in current localization. Patrón por defecto USA: "dddd, dd MMMM yyyy"
<code>vk pattern long time</code>	<code>"longTimePattern"</code>	ISO 8601 format for the time in current localization. USA default pattern: "HH:mm:ss"
<code>vk pattern month day</code>	<code>"monthDayPattern"</code>	ISO 8601 format for the month and day in current localization. USA default pattern: "MMMM dd"
<code>vk pattern short date</code>	<code>"shortDatePattern"</code>	Abbreviated ISO 8601 format for the date in current localization. USA default pattern: "MM/dd/yyyy"
<code>vk pattern short time</code>	<code>"shortTimePattern"</code>	Abbreviated ISO 8601 format for the time in current localization. Patrón por defecto USA: "HH:mm"
<code>vk pattern sortable date time</code>	<code>"sortableDateTimePattern"</code>	ISO 8601 format for the date and time in current localization which can be sorted. USA default pattern: "yyyy'-'MM'-'dd'T'HH':'mm':'ss"
<code>vk pattern universal sortable date time</code>	<code>"universalSortableDateTimePattern"</code>	ISO 8601 format for the date and time in current localization using UTC which can be sorted. USA default pattern: "yyyy'-'MM'-'dd HH':'mm':'ss'Z"
<code>vk pattern year month</code>	<code>"yearMonthPattern"</code>	ISO 8601 format for the month and year in current localization. Patrón por defecto USA: "yyyy MMMM"

## Ejemplo

```
//Definir el valor de la celda como fecha y hora específicas
VP SET VALUE(VP Cell("ViewProArea";3;9);New object("value";!2024-12-18!);"time";?14:30:10?;"format";vk p
```

## Formato fecha y hora personalizados

To create your own date and time patterns, in your current localization, you can use combinations of the following codes:

	Code (no distingue entre mayúsculas y minúsculas)	Descripción	Ejemplo
Fecha			(January 1, 2019)
	m	Month number without leading zero	1
	mm	Número de mes con cero precedente	01
	mmm	Nombre del mes, corto	Jan
	mmmm	Nombre del mes, long	January
	d	Day number without leading zero	1
	dd	Número de días con cero precedente	01
	ddd	Día de la semana, corto	Tue
	dddd	Día de la semana, largo	Tuesday
	yy	Año, formato corto	19
	yyyy	Año, formato largo	2019
Hora			(2:03:05 PM)
	h	Hora sin cero precedente. 0-23	2
	hh	Hora con cero precedente. 00-23	02
	m	Minutes without leading zero. 0-59	3
	mm	Minutos con cero precedente. 00-59	03
	s	Seconds without leading zero. 0-59	5
	ss	Segundo con cero precedente. 00-59	05
	[h]	Tiempo transcurrido en horas	14 (puede superar 24)
	[mm]	Tiempo transcurrido en minutos	843
	[ss]	Tiempo transcurrido en segundos	50585
	AM/PM	Periodos del día. Se utiliza el formato de 24 horas si se omite.	PM

The code 'm' is interpreted depending on its position in the pattern. If it's immediately after 'h' or 'hh' or immediately before 's' or 'ss', it will be interpreted as minutes, otherwise it will be interpreted as months.

## Símbolos adicionales

In addition to the special characters and codes described in the previous sections, there are additional characters and symbols that can be used in your format patterns. These additional characters and symbols do not require a \$ or "" and do not impact the interpretation of the format pattern. They appear "as-is" within the pattern.

Carácter	Descripción	Ejemplo
+ y -	Signos más y menos	### + ### = ###,###
( )	Paréntesis izquierdo y derecho	(-###.###)
:	Dos puntos	hh:mm:ss
^	Caret	#^#
'	Apostrofe	'#####
{ }	Paréntesis curvos	{###,###,###}
< >	Signos menor que y mayor que	## >##
=	Signo igual	#+=##
/	Barra inclinada hacia adelante. When used with numbers, displays them as fractions.	mm/dd/yyyy
!	Signo de exclamación	\$###.00!
&	Ampersand	"Hello" & "Welcome"
~	Tilde	~##
	Carácter de espacio	
€	Euro	€###.00
£	Libra esterlina	£###.00
¥	Yen japonés	¥###.00
\$	Signo dólar	\$###.00
¢	Cent sign	.00¢

## Atributos de impresión

4D View Pro print attributes allow you to control all aspects of printing 4D View Pro areas. These attributes are handled by the following commands:

- [VP SET PRINT INFO](#)
- [VP Get print info](#)

## Columnas / Líneas

Column and row attributes are used to specify the beginning, end, and repetition of columns and rows.

Propiedad	Tipo	Descripción
columnEnd	entero largo	The last row to print in a cell range. Default value = -1 (all rows)
columnStart	entero largo	The first row to print in a cell range. Default value = -1 (all rows)
repeatColumnEnd	entero largo	The last column of a range of columns to print on the left of each page. Default value = -1 (all rows)
repeatColumnStart	entero largo	The first column of a range of columns to print on the left of each page. Default value = -1 (all rows)
repeatRowEnd	entero largo	The last row of a range of rows to print on the top of each page. Default value = -1 (all rows)
repeatRowStart	entero largo	The first row of a range of rows to print at the top of each page. Default value = -1 (all rows)
rowEnd	entero largo	The last column to print in a cell range. Default value = -1 (all rows)
rowStart	entero largo	The first column to print in a cell range. Default value = -1 (all rows)

## Encabezados / Pies de página

Header and footer attributes are used to specify text or images in the left, right, and center header/footer sections.

Propiedad	Tipo	Descripción
footerCenter	texto	The text and format of the center footer on printed pages.
footerCenterImage	picture   text*	La imagen para la sección central del pie de página.
footerLeft	texto	The text and format of the left footer on printed pages.
footerLeftImage	picture   text*	La imagen de la parte izquierda del pie de página.
footerRight	texto	The text and format of the right footer on printed pages.
footerRightImage	picture   text*	La imagen de la parte derecha del pie de página.
headerCenter	texto	The text and format of the center header on printed pages.
headerCenterImage	picture   text*	La imagen para la sección central del encabezado.
headerLeft	texto	The text and format of the left header on printed pages.
headerLeftImage	picture   text*	La imagen de la sección izquierda del encabezado.
headerRight	texto	The text and format of the right header on printed pages.
headerRightImage	picture   text*	La imagen de la sección derecha del encabezado.

\* If using text type, pass the filepath (absolute or relative) of the image. If you pass a relative path, the file should be located next to the database structure file. En Windows, la extensión del archivo debe ser indicada. No matter the type used to set an image, the image itself (not a reference) is stored in the 4D View Pro area and is returned by [VP Get print info](#).

## Caracteres especiales

The following special characters allow the automatic addition or formatting of information in the header and footer when the 4D View Pro area is printed.

Carácter	Descripción	Ejemplo	Resultado
&	Escape character	(ver ejemplos más abajo)	
P	Página actual	printInfo.headerLeft:="Esta es la página &P."	Esta es la página 5.
N	Conteo de páginas	printInfo.headerLeft:="Hay &N páginas."	Hay 10 páginas.
D	Fecha actual (formato yyyy/mm/dd)	printInfo.headerLeft:="Es &D."	Es 2015/6/19.
T	Hora actual	printInfo.headerLeft:="Es &T."	Es 16:30:36.
G	Imagen	printInfo.headerLeftImage:=smiley printInfo.headerLeft:="&G"	
S	Strikethrough	printInfo.headerLeft:="&SEsto es texto."	<del>Esto es texto.</del>
U	Subrayado	printInfo.headerLeft:="&UEsto es texto."	<u>Esto es texto.</u>
B	Negrita	printInfo.headerLeft:="&BEsto es texto."	Esto es texto.
I	Itálica	printInfo.headerLeft:="&IEsto es texto."	<i>Esto es texto.</i>
"	Prefijo fuente	printInfo.headerLeft:="&"Lucida Console"&14This is text."	<b>This is text.</b>
K	Prefijo de color de texto	printInfo.headerLeft:="&KFF0000This is text."	<b>Esto es texto.</b>
F	Nombre del libro	printInfo.headerLeft:="&F"	2019 Monthly Revenue Forecasts
A	Nombre de la hoja de cálculo	printInfo.headerLeft:="&A"	June 2019 revenue forecast

## Márgenes

Margin attributes are used to specify the 4D View Pro area margins for printing. Expressed in hundreds of an inch.

Propiedad		Tipo	Descripción
margin		objeto	Los márgenes de impresión
	top	entero largo	Margen superior, en centésimas de pulgada. Por defecto = 75
	bottom	entero largo	El margen inferior, en centésimas del pulgada. Por defecto = 75
	left	entero largo	Margen derecho, en centésimas de pulgada. Por defecto = 70
	right	entero largo	Margen izquierdo, en centésimas de pulgada. Por defecto = 70
	header	entero largo	Header offset, in hundredths of an inch. Por defecto = 30
	footer	entero largo	Footer offset, in hundredths of an inch. Por defecto = 30

## Orientación

Orientation attributes are used to specify the direction the printed page layout.

Este atributo define sólo la información de renderizado.

Propiedad	Tipo	Descripción
orientation	entero largo	Orientación de la página. Valores disponibles: <code>vk print page orientation landscape</code> , <code>vk print page orientation portrait</code> (por defecto)

## Página

Page attributes are used to specify general document print settings.

Propiedad	Tipo	Descripción
blackAndWhite	booleano	Printing in black and white only. Valor por defecto: false  Nota: los PDFs no se ven afectados por este atributo. Los colores en PDFs permanecen.
centering	entero largo	Como se centran los contenidos en la página impresa. Valores disponibles: <code>vk print centering both</code> , <code>vk print centering horizontal</code> , <code>vk print centering none</code> (por defecto), <code>vk print centering vertical</code>
firstPageNumber	entero largo	El número de página a imprimir en la primera página. Por defecto = 1
pageOrder	entero largo	Las páginas del pedido se imprimen. Valores disponibles: <code>vk print page order auto</code> (por defecto), <code>vk print page order down then over</code> , <code>vk print page order over then down</code> .
pageRange	texto	The range of pages for printing
qualityFactor	entero largo	El factor de calidad para la impresión (1 - 8). The higher the quality factor, the better the printing quality, however printing performance may be affected. Por defecto = 2
useMax	booleano	Sólo se imprimen columnas y líneas con datos. Valor por defecto: true
zoomFactor	real	La cantidad para ampliar o reducir la página impresa. Por defecto = 1

## Tamaño del papel

Paper size attributes are used to specify the dimensions or model of paper to use for printing. Hay dos maneras de definir el tamaño del papel:

- Tamaño personalizado - atributos de alto y ancho
- Tamaño estándar - atributo kind

Propiedad		Tipo	Descripción
paperSize		objeto	Paper dimensions (height, width) or specific format (kind) for printing.
	height	entero largo	La altura del papel, en centésimas de pulgada.
	ancho	entero largo	Ancho del papel, en centésimas de pulgada.
	kind	texto	Name of standard paper size (e.g., A2, A4, legal, etc.) returned by <code>Get Print Option</code> . Valor por defecto: "letter"

## Escala

Scale attributes are used to specify printing optimization and adjustments.

Propiedad	Tipo	Descripción
bestFitColumns	booleano	Column width is adjusted to fit the largest text width for printing. Valor por defecto: "false"
bestFitRows	booleano	Row height is adjusted to fit the tallest text height for printing. Valor por defecto: "false"
fitPagesTall	entero largo	The number of vertical pages (portrait orientation) to check when optimizing printing. Por defecto = -1
fitPagesWide	entero largo	The number of horizontal pages (landscape orientation) to check when optimizing printing. Por defecto = -1

## Mostrar / Ocultar

Show / Hide attributes are used to specify the visibility (printing) of 4D View Pro area elements.

Propiedad	Tipo	Descripción
showBorder	booleano	Imprime el borde del contorno. Valor por defecto: "true"
show.ColumnHeader	entero largo	Column header print settings. Valores disponibles: <code>vk print visibility hide</code> , <code>vk print visibility inherit</code> (por defecto), <code>vk print visibility show</code> , <code>vk print visibility show once</code>
show.GridLine	booleano	Imprime las líneas de la cuadrícula. Valor por defecto: "false"
show.RowHeaders	entero largo	Row headers print settings. Valores disponibles: <code>vk print visibility hide</code> , <code>vk print visibility inherit</code> (por defecto), <code>vk print visibility show</code> , <code>vk print visibility show once</code>

## Marca de agua

Watermark attributes are used to superimpose text or an image onto the 4D View Pro area.

Propiedad		Tipo	Descripción
marca de agua		colección	Collection of watermark settings. Valor por defecto: indefinido
	[ ].height	entero largo	The height of the watermark text / image.
	[ ].imageSrc	picture   text*	The watermark text / image.
	[ ].page	texto	La(s) página(s) donde se imprime la marca de agua. Para todas las páginas: "all". For specific pages: page numbers or page ranges separated by commas. Ej.: "1,3,5-12"
	[ ].width	entero largo	El ancho del texto/imagen marca de agua.
	[ ].x	entero largo	The horizontal coordinate of the top left point of the watermark text / image.
	[ ].y	entero largo	The vertical coordinate of the top left point of the watermark text / image.

\* If using text type, pass the filepath (absolute or relative) of the image. If you pass a relative path, the file should be located next to the database structure file. En Windows, la extensión del archivo debe ser indicada. No matter the type used to set an image, the image itself (not a reference) is stored in the 4D View Pro area and is returned by [VP Get print info](#).

## Style Objects

4D View Pro style objects and style sheets allow you to control the graphical aspects and the look of your 4D View Pro documents.

### Objetos de estilo & Hojas de estilo

Style objects contain the style settings. They can be used either in a style sheet or on their own. Style objects can also be used in addition to a style sheet so that different settings can be specified for individual cell ranges without affecting the rest of the document. You can use style objects directly with the [VP SET CELL STYLE](#) and [VP SET DEFAULT STYLE](#) commands.

A style sheet groups together a combination of properties in a style object to specify the look of all of the cells in your 4D View Pro documents. Style sheets saved with the document can be used to set the properties for a single sheet, multiple sheets, or an entire workbook. When created, a 4D View Pro style sheet is given a name which is saved within the style sheet in the "name" property. This allows a style sheet to be easily used and, if thoughtfully selected, can facilitate its identification and purpose (e.g., Letterhead\_internal, Letterhead\_external).

Style sheets are created with the [VP ADD STYLESHEET](#) command and applied with the the [VP SET DEFAULT STYLE](#) or [VP SET CELL STYLE](#) commands. You can remove a style sheet with the [VP REMOVE STYLESHEET](#) command.

The [VP Get stylesheet](#) command can be used to return the style object of a single style sheet or you can use the [VP Get stylesheets](#) command to retrieve a collection of style objects for multiple style sheets.

### Propiedades del objeto de estilo

Ejemplo:

```

$style:=New object
$style.hAlign:=vk horizontal align left
$style.font:="12pt papyrus"
$style.backColor:="#E6E6FA" //color morado claro

VP SET DEFAULT STYLE("myDoc";$style)

```

## Background & Foreground

Propiedad	Tipo	Descripción	Valores posibles
backColor	texto	Define el color del fondo.	CSS color "#rrggbb" syntax (preferred syntax), CSS color "rgb(r,g,b)" syntax (alternate syntax), CSS color name (alternate syntax)
backgroundImage	picture, text	Especifica una imagen de fondo.	Can be specified directly or via the image path (full path or file name only). If the file name only is used, the file must be located next to the database structure file. No matter how set (picture or text), a picture is saved with the document. This could impact the size of a document if the image is large. Nota para Windows: la extensión del archivo incluirse.
backgroundImageLayout	entero largo	Define el diseño para la imagen de fondo.	vk image layout center, vk image layout none, vk image layout stretch, vk image layout zoom
foreColor	texto	Define el color del primer plano.	CSS color "#rrggbb" syntax (preferred syntax), CSS color "rgb(r,g,b)" syntax (alternate syntax), CSS color name (alternate syntax)

## Bordes

Propiedad		Tipo	Descripción	Valores posibles
borderBottom, borderLeft, borderRight, borderTop, diagonalDown, diagonalUp		objeto	Define la línea de borde correspondiente	
	color	texto	Define el color del borde. Por defecto = black.	CSS color "#rrggbb" syntax (preferred syntax), CSS color "rgb(r,g,b)" syntax (alternate syntax), CSS color name (alternate syntax)
	style	entero largo	Define el estilo del borde. Por defecto = empty. No puede ser null o indefinido.	vk line style dash dot, vk line style dash dot dot, vk line style dashed, vk line style dotted, vk line style double, vk line style empty, vk line style hair, vk line style medium, vk line style medium dash dot, vk line style medium dash dot dot, vk line style medium dashed, vk line style slanted dash dot, vk line style thick

## Fuentes y texto

Propiedad		Tipo	Descripción	Valores posibles
font		texto	Specifies the font characteristics in CSS font shorthand ("font-style font-variant font-weight font-size/line-height font-	A CSS font shorthand. 4D provides utility commands to handle

Propiedad		Tipo	font-size, font-family Example: "14pt Century Gothic".  The font-size and font-family values are mandatory. If one of the other values is missing, their default values are used. Note: If a font name contains a space, the name must be within quotes.	font characteristics as objects: <a href="#">VP Font to object</a> and <a href="#">VP Object to font</a>
formatter		texto	Patrón de propiedad valor/tiempo.	Number/text/date/time formats, special characters. Ver <a href="#">Formato de celda</a> .
isVerticalText		booleano	Especifica la dirección del texto.	True = texto vertical, False = texto horizontal.
labelOptions		objeto	Define las opciones de etiqueta de celda (opciones de marca de agua).	
	alignement	entero largo	Especifica la posición de la etiqueta de la celda. Propiedad opcional.	<code>vk label alignment top left, vk label alignment bottom left, vk label alignment top center, vk label alignment bottom center, vk label alignment top right, vk label alignment bottom right</code>
	visibility	entero largo	Especifica la visibilidad de la etiqueta de la celda. Propiedad opcional.	<code>vk label visibility auto, vk label visibility hidden, vk label visibility visible</code>
	foreColor	texto	Define el color del primer plano. Propiedad opcional.	CSS color "#rrggbb" syntax (preferred syntax), CSS color "rgb(r,g,b)" syntax (alternate syntax), CSS color name (alternate syntax)
	font	texto	Specifies the font characteristics with CSS font shorthand ("font-style font-variant font-weight font-size/line-height font-family"). The font-size and font-family values are mandatory.	
textDecoration		entero largo	Especifica la decoración añadida al texto.	<code>vk text decoration double underline, vk text decoration line through, vk text decoration none, vk text decoration overline, vk text decoration underline</code>
textIndent		entero largo	Define la unidad de indentación del texto. 1 = 8 píxeles	
textOrientation		entero largo	Define el ángulo de rotación del texto en una celda. Número entre -90 y 90	

Marca de agua		Tipo	Define el contenido de la marca de agua (etiqueta de la celda)	Valores posibles
wordWrap		booleano	Especifica si el texto debe ser ajustado.	True = texto ajustado, False = texto no ajustado

## Disposición

Propiedad	Tipo	Descripción	Valores posibles
cellPadding	texto	Define el relleno de la celda	
hAlign	entero largo	Define la alineación horizontal del contenido de la celda.	<code>vk horizontal align center, vk horizontal align general, vk horizontal align left, vk horizontal align right</code>
locked	booleano	Specifies cell protection status. Note, this is only available if <a href="#">sheet protection</a> is enabled.	True = bloqueado, False = desbloqueado.
shrinkToFit	booleano	Specifies if the contents of the cell should be reduced.	True = contenido reducido, False = sin reducción.
tabStop	booleano	Specifies if the focus to the cell can be set using the Tab key.	True = Tab key sets focus, False = Tab key does not set focus.
vAlign	entero largo	Especifica la alineación vertical del contenido de la celda.	<code>vk vertical align bottom, vk vertical align center, vk vertical align top</code>

## Información de estilo

Propiedad	Tipo	Descripción
name	texto	Define el nombre del estilo
parentName	texto	Specifies the style that the current style is based on. Values from the parent style will be applied, then any values from the current style are applied. Changes made in the current style will not be relected in the parent style. Sólo está disponible cuando se utiliza una hoja de estilo.

## Objeto 4D View Pro

The 4D View Pro [object](#) stores the whole spreadsheet contents. Es manejado automáticamente por 4D View Pro. You can set or get this object using the [VP IMPORT FROM OBJECT](#) or [VP Export to object](#) methods.

Contiene las siguientes propiedades:

Propiedad	Tipo de valor	Descripción
version	Entero largo	Versión del componente interno
dateCreation	Timestamp	Fecha de creación
dateModified	Timestamp	Fecha última modificación
meta	Objeto	Contenido gratuito, reservado para el desarrollador 4D
spreadJS	Objeto	Reservado para el componente 4D View Pro

## 4D View Pro Form Object Variable

The 4D View Pro form object variable is the [object](#) variable associated to the 4D View Pro form area. It manages

information used by the 4D View Pro object.

The 4D View Pro form object variable is for information purposes only (i.e., debugging). Bajo ninguna circunstancia debe modificarse.

Contiene las siguientes propiedades:

Propiedad	Tipo de valor	Descripción
ViewPro.area	Texto	Nombre del área 4D View Pro
ViewPro.callbacks	Objeto	Stores temporary information necessary for commands requiring callbacks such as importing and exporting.
ViewPro.commandBuffers	Collection	Stores sequentially the commands called by the method and executes them as a batch (rather than individually) upon exiting the method, or if a command returns a value or the <a href="#">VP FLUSH COMMANDS</a> is called. This mechanism increases performance by reducing the number of requests sent.
ViewPro.events	Objeto	<a href="#">Event</a> list.
ViewPro.formulaBar	Booleano	Indicates whether or not the formula bar is displayed. Disponible sólo para la interfaz de la "barra de herramientas".
ViewPro.initialized	Booleano	Indicates whether or not the 4D View Pro area has been initialized (see <a href="#">On VP Ready</a> event).
ViewPro.interface	Texto	Specifies the type of user interface:"ribbon", "toolbar", "none".

# Fórmulas y funciones

## Utilización de las fórmulas

A spreadsheet formula is an expression that calculates the value of a cell.

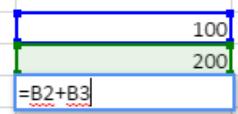
### Entrada de las fórmulas

Para introducir una fórmula en un área 4D View Pro:

1. Select the cell into which you will enter the formula or function.
2. Introduzca = (el signo igual).
3. Type the formula and hit the Enter key.

When writing a formula, you can use different shortcuts:

- click on a cell to enter its reference in the formula:



- escriba la primera letra de una función a ingresar. A pop-up menu listing the available functions and references appears, allowing you to select the desired elements:

You can also create named formulas that can be called via their name. To do so, enter these formulas using the [VP ADD FORMULA NAME](#) command.

### Operadores y operandos

Todas las fórmulas tienen operandos y operadores:

- Operators: see [Values and operators](#) below.
- Operands include several categories:
  - [values](#) (5 data types are supported)
  - [references to other cells](#) (relative, absolute, mixed or by name)
  - [funciones estándar de hoja de cálculo](#)
  - [4D functions](#) based upon 4D formulas and providing access to 4D variables, fields, methods, commands, or expressions.

### Valores y operadores

4D View Pro soporta cinco tipos de datos. For each data type, specific literal values and operators are supported.

Tipos de datos	Valores	Operadores
Número	1.2 1.2 E3 1.2E-3 10.3x	+ (addition) - (subtraction) * (multiplication) / (division) ^ (exponent, the number of times to multiply a number by itself) % (percentage -- divide the number before the operator by one hundred)
Fecha	10/24/2017	+ (date + number of days -> date) + (date + time -> date + time of day) - (date - number of days -> date) - (date - date -> number of days between the two)
Hora	10:12:10	Operadores de duración: + (addition) - (subtraction) * (duration * number -> duration) / (duration / number -> duration)
Cadena	'Sophie' o "Sophie"	& (concatenación)
Booleano	TRUE o FALSE	-

## Operadores de comparación

The following operators can be used with two operands of the same type:

Operador	Comparación
=	igual a
<>	es diferente de
>	mayor que
<	menor que
>=	mayor o igual que
<=	menor o igual que

## Presedencia de los operadores

Lista de los operadores de la mas a menos importante:

Operador	Descripción
()	Paréntesis (para agrupar)
-	Negativo
+	Más
%	Porcentaje
^	Exponente
* y /	Multiplicar y dividir
+ y -	Add and Subtract
&	Concatenar
= > < >= <= <>	Comparar

## Referencias de celdas

Formulas often refer to other cells by cell addresses. Puede copiar estas fórmulas a otras celdas. For example, the following formula, entered in cell C8, adds the values in the two cells above it and displays the result.

= C6 + C7

Esta fórmula se refiere a las celdas C6 y C7. That is, 4D View Pro is instructed to refer to these other cells for values to use in the formula.

When you copy or move these formulas to new locations, each cell address in that formula will either change or stay the same, depending on how it is typed.

- A reference that changes is called a relative reference, and refers to a cell by how far left/right and up/down it is from the cell with the formula.
- A reference that always points to a particular cell is called an absolute reference.
- You can also create a mixed reference which always points to a fixed row or column.

### Notación de las referencias

If you use only cell coordinates, for example, `C5`, 4D View Pro interprets the reference as relative. You may make the reference an absolute reference by putting a dollar sign in front of the letter and the number, as in `$C$5`.

You can mix absolute and relative references by inserting a dollar sign in front of the letter or the number alone, for example, `$C5` or `C$5`. A mixed reference allows you to specify either the row or the column as absolute, while allowing the other portion of the address to refer relatively.

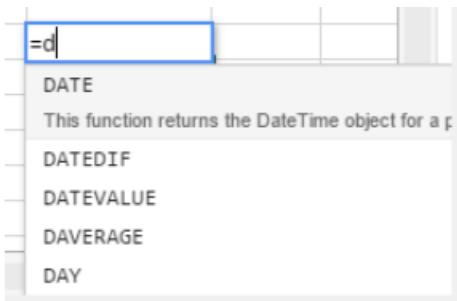
A convenient, fast and accurate way to specify an absolute reference is to name the cell and use that name in place of the cell address. Una referencia a una celda nombrada es siempre absoluta. You can create or modify named cells or named cell ranges using the `VP ADD RANGE NAME` method.

The following table shows the effect of the different notations:

Ejemplo	Tipo de referencia	Descripción
C5	Relativo	Reference is to the relative location of cell C5, depending on the location of the cell in which the reference is first used
\$C\$5	Absolute	La referencia es absoluta. Will always refer to cell C5 no matter where it is used.
\$C5	Mixed	Reference is always to column C, but the row reference is relative to the location of the cell in which the reference is first used.
C\$5	Mixed	Reference is always to row 5, but the column reference is relative to the location of the cell in which the reference is first used
Nombre de la celda	Absolute	La referencia es absoluta. Will always refer to the <a href="#">named cell or range</a> no matter where the reference is used.

## Funciones integradas

Spreadsheet functions are preset formulas used to calculate cell values. When you type the first letter of the function to enter, a pop-up menu listing the available functions and references appears, allowing you to select the desired elements:



See [SpreadJS's extended list of functions](#) for details and examples.

## Funciones 4D

4D View Pro allows you to define and call 4D custom functions, which execute [4D formulas](#). Using 4D custom functions extends the possibilities of your 4D View Pro documents and allows powerful interactions with the 4D database.

4D custom functions provide access, from within your 4D View Pro formulas, to:

- variables proceso 4D,
- campos,
- project methods,
- comandos del lenguaje 4D,
- o cualquier expresión válida de 4D.

4D custom functions can receive [parameters](#) from the 4D View Pro area, and return values.

You declare all your functions using the [VP SET CUSTOM FUNCTIONS](#) method. Ejemplos:

```
o:=New object

//Name of the function in 4D View Pro: "DRIVERS_LICENCE"
$o.DRIVERS_LICENCE:=New object

//process variable
$o.DRIVERS_LICENCE.formula:=Formula(DriverLicence)

//table field
$o.DRIVERS_LICENCE.formula:=Formula([Users]DriverLicence)

//project method
$o.DRIVERS_LICENCE.formula:=Formula(DriverLicenceState)

//4D command
$o.DRIVERS_LICENCE:=Formula(Choose(DriverLicence; "Obtained"; "Failed"))

//4D expression and parameter
$o.DRIVERS_LICENCE.formula:=Formula(ds.Users.get($1).DriverLicence)
$o.DRIVERS_LICENCE.parameters:=New collection
$o.DRIVERS_LICENCE.parameters.push(New object("name"; "ID"; "type"; Is longint))
```

See also [4D View Pro: Use 4D formulas in your spreadsheet \(blog post\)](#)

## Ejemplo con Hello World

We want to print "Hello World" in a 4D View Pro area cell using a 4D project method:

1. Create a "myMethod" project method with the following code:

```
#DECLARE->$hw Text  
$hw:="Hello World"
```

2. Execute the following code before opening any form that contains a 4D View Pro area:

```
Case of  
:(Form event code=On Load)  
  var $o : Object  
  $o:=New object  
  // Define "vpHello" function from the "myMethod" method  
  $o.vpHello:=New object  
  $o.vpHello.formula:=Formula(myMethod)  
  VP SET CUSTOM FUNCTIONS("ViewProArea";$o)  
End case
```

3. Edit the content of a cell in a 4D View Pro area and type:



"myMethod" is then called by 4D and the cell displays:



## Parámetros

Parameters can be passed to 4D functions that call project methods using the following syntax:

```
=METHODNAME(param1,param2,...,paramN)
```

These parameters are received in *methodName* in \$1, \$2...\$N.

Note that the ( ) are mandatory, even if no parameters are passed:

```
=METHODWITHOUTNAME()
```

You can declare the name, type, and number of parameters through the *parameters* collection of the function you declared using the [VP SET CUSTOM FUNCTIONS](#) method. Optionally, you can control the number of parameters passed by the user through *minParams* and *maxParams* properties.

For more information on supported incoming parameter types, please refer to the [VP SET CUSTOM FUNCTIONS](#) method description.

If you do not declare parameters, values can be sequentially passed to methods (they will be received in \$1, \$2...) and their type will be automatically converted. Dates in *jstype* will be passed as [object](#) in 4D code with two properties:

Property	Type	Description	--- --- ---	value	Date	Date value	time	Real	Time in seconds
----------	------	-------------	-------------	-------	------	------------	------	------	-----------------

4D project methods can also return values in the 4D View Pro cell formula via \$0. The following data types are supported for returned parameters:

- [text](#) (converted to string in 4D View Pro)
- [real/longint](#) (converted to number in 4D View Pro)
- [date](#) (converted to JS Date type in 4D View Pro - hour, minute, sec = 0)

- **time** (converted to JS Date type in 4D View Pro - date in base date, i.e. 12/30/1899)
- **boolean** (converted to bool in 4D View Pro)
- **picture** (jpg,png,gif,bmp,svg other types converted into png) creates a URI (data:image/png;base64,xxxx) and then used as the background in 4D View Pro in the cell where the formula is executed
- **object** with the following two properties (allowing passing a date and time):

Propiedad	Tipo	Descripción
value	Fecha	Valor fecha
time	Real	Hora en segundos

If the 4D method returns nothing, an empty string is automatically returned.

Se devuelve un error en la celda 4D View Pro si:

- the 4D method returns another type other than those listed above,
- an error occurred during 4D method execution (when user clicks on "abort" button).

## Ejemplo

```
var $o : Object

$o.BIRTH_INFORMATION:=New object
$o.BIRTH_INFORMATION.formula:=Formula(BirthInformation)
$o.BIRTH_INFORMATION.parameters:=New collection
$o.BIRTH_INFORMATION.parameters.push(New object("name";"First name";"type";Is text))
$o.BIRTH_INFORMATION.parameters.push(New object("name";"Birthday";"type";Is date))
$o.BIRTH_INFORMATION.parameters.push(New object("name";"Time of birth";"type";Is time))
$o.BIRTH_INFORMATION.summary:="Returns a formatted string from given information"

VP SET CUSTOM FUNCTIONS("ViewProArea"; $o)
```

□

## Compatibilidad

Alternate solutions are available to declare fields or methods as functions in your 4D View Pro areas. These solutions are maintained for compatibility reasons and can be used in specific cases. However, using the `VP SET CUSTOM FUNCTIONS` method is recommended.

## Referenciación de campos mediante la estructura virtual

4D View Pro allows you to reference 4D fields using the virtual structure of the database, i.e. declared through the `SET TABLE TITLES` and/or `SET FIELD TITLES` commands with the \* parameter. This alternate solution could be useful if your application already relies on a virtual structure (otherwise, using `VP SET CUSTOM FUNCTIONS` is recommended).

**WARNING:** You cannot use the virtual structure and `VP SET CUSTOM FUNCTIONS` simultaneously. As soon as `VP SET CUSTOM FUNCTIONS` is called, the functions based upon `SET TABLE TITLES` and `SET FIELD TITLES` commands are ignored in the 4D View Pro area.

## Requisitos

- The field must belong to the virtual structure of the database, i.e. it must be declared through the `SET TABLE TITLES` and/or `SET FIELD TITLES` commands with the \* parameter (see example),
- Table and field names must be ECMA compliant (see [ECMA Script standard](#)),
- The field type must be supported by 4D View Pro (see above).

An error is returned in the 4D View Pro cell if the formula calls a field which is not compliant.

## Llamar a un campo virtual en una fórmula

To insert a reference to a virtual field in a formula, enter the field with the following syntax:

```
TABLENAME_FIELDNAME()
```

For example, if you declared the "Name" field of the "People" table in the virtual structure, you can call the following functions:

```
=PEOPLE_NAME()  
=LEN(PEOPLE_NAME())
```

If a field has the same name as a [4D method], it takes priority over the method.

## Ejemplo

We want to print the name of a person in a 4D View Pro area cell using a 4D virtual field:

1. Crear una tabla "Employee" con un campo "L\_Name":

Employee	
ID	2 <sup>32</sup>
L_Name	A
F_Name	A

2. Execute the following code to initialize a virtual structure:

```
ARRAY TEXT($tableTitles;1)  
ARRAY LONGINT($tableNum;1)  
$tableTitles{1}:="Emp"  
$tableNum{1}:=2  
SET TABLE TITLES($tableTitles;$tableNum;*)  
  
ARRAY TEXT($fieldTitles;1)  
ARRAY LONGINT($fieldNum;1)  
$fieldTitles{1}:="Name"  
$fieldNum{1}:=2 //last name  
SET FIELD TITLES([Employee];$fieldTitles;$fieldNum;*)
```

3. Edit the content of a cell in the 4D View Pro area and enter "=e":

	A	B	C	D	E
1	=e				
2	EDATE				
3	EFFECT				
4	EMP_NAME				
5	ENCODEURL				

4. Select EMP\_NAME (use the Tab key) and enter the closing ).

	A	B
1	=EMP_NAME()	

5. Validate the field to display the name of the current employee:

	A	B
1	Smith	

La tabla [Employee] debe tener un registro actual.

## Declarar los métodos autorizados

You can call directly 4D project methods from within your 4D View Pro formulas. For security reasons, you must declare explicitly methods that can be called by the user with the [VP SET ALLOWED METHODS](#) method.

### Requisitos

To be called in a 4D View Pro formula, a project method must be:

- Allowed: it was explicitly declared using the [VP SET ALLOWED METHODS](#) method.
- Runnable: it belongs to the host project or a loaded component with the "Shared by components and host project" option enabled (see [Sharing of project methods](#)).
- Not in conflict with an existing 4D View Pro spreadsheet function: if you call a project method with the same name as a 4D View Pro built-in function, the function is called.

If neither the [VP SET CUSTOM FUNCTIONS](#) nor the [VP SET ALLOWED METHODS](#) method has been executed during the session, 4D View Pro custom functions rely on allowed methods defined by 4D's generic [SET ALLOWED METHODS](#) command. In this case, the project method names must comply with JavaScript Identifier Grammar (see [ECMA Script standard](#)). The global filtering option in the Settings dialog box (see [Data Access](#)) is ignored in all cases.

# Lista de los métodos

Warning: The commands on this page are not thread-safe.

A - C - D - E - F - G - I - M - N - O - P - R - S

## A

### VP ADD FORMULA NAME

VP ADD FORMULA NAME ( *vpAreaName* : Text ; *vpFormula* : Text ; *name* : Text { ; *options* : Object } )

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
<i>vpFormula</i>	Texto	->	Fórmula 4D View Pro
<i>name</i>	Texto	->	Nombre de la fórmula
<i>options</i>	Objeto	->	Opciones de la fórmula nombrada

#### Descripción

The `VP ADD FORMULA NAME` command creates or modifies a named formula in the open document.

Named formulas created by this command are saved with the document.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

Pass the 4D View Pro formula that you want to name in *vpFormula*. For detailed information about formula syntax, see [Formulas and Functions](#) page.

Pass the new name for the formula in *name*. If the name is already used within the same scope, the new named formula replaces the existing one. Note that you can use the same name for different scopes (see below).

You can pass an object with additional properties for the named formula in *options*. Se soportan las siguientes propiedades:

Propiedad	Tipo	Descripción
<i>scope</i>	Número	Alcance de la fórmula. Puede pasar el índice de la hoja (la numeraciòn comienza en 0) o utilizar las siguientes constantes: <ul style="list-style-type: none"><li>• <code>vk current sheet</code></li><li>• <code>vk workbook</code></li></ul> El alcance determina si el nombre de una fórmula es local para una hoja de cálculo determinada ( <i>scope=sheet index</i> or <code>vk current sheet</code> ), o si es global a todo el libro de trabajo ( <code>scope= vk workbook</code> ).
<i>comment</i>	Texto	Comentario asociado a una fórmula nombrada

#### Ejemplo

```
VP ADD FORMULA NAME("ViewProArea";"SUM($A$1:$A$10)";"Total2")
```

## Ver también

[Cell references](#)

[VP ADD RANGE NAME](#)

[VP Get formula by name](#)

[VP Get names](#)

## VP ADD RANGE NAME

VP ADD RANGE NAME ( *rangeObj* : Object ; *name* : Text { ; *options* : Object } )

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Texto	->	Objeto rango
<i>name</i>	Texto	->	Nombre de la fórmula
<i>options</i>	Objeto	->	Opciones de la fórmula nombrada

### Descripción

The `VP ADD RANGE NAME` command creates or modifies a named range in the open document.

Named ranges created by this command are saved with the document.

In *rangeObj*, pass the range that you want to name and in *name*, pass the new name for the range. If the name is already used within the same scope, the new named range replaces the existing one. Note that you can use the same name for different scopes (see below).

You can pass an object with additional properties for the named range in *options*. Se soportan las siguientes propiedades:

Propiedad	Tipo	Descripción
<i>scope</i>	Número	Alcance del rango. You can pass the sheet index (counting begins at 0) or use the following constants: <ul style="list-style-type: none"><li>• <code>vk current sheet</code></li><li>• <code>vk workbook</code></li></ul> El alcance determina si el nombre de un rango es local para una hoja de cálculo determinada ( <i>scope=sheet index</i> or <code>vk current sheet</code> ), o si es global a todo el libro de trabajo ( <i>scope= vk workbook</i> ).
<i>comment</i>	Texto	Comentario asociado al rango nombrado

- A named range is actually a named formula containing coordinates. `VP ADD RANGE NAME` facilitates the creation of named ranges, but you can also use the `VP ADD FORMULA NAME` method to create named ranges.
- Formulas defining named ranges can be retrieved with the `VP Get formula by name` method.

## Ejemplo

You want to create a named range for a cell range:

```
$range:=VP Cell("ViewProArea";2;10)
VP ADD RANGE NAME($range;"Total1")
```

## Ver también

[VP ADD FORMULA NAME](#)

[VP Get formula by name](#)

VP Get names

VP Name

## VP ADD SELECTION

VP ADD SELECTION ( *rangeObj* : Object )

Parámetros	Tipo		Descripción
rangeObj	Texto	->	Objeto rango

### Descripción

The **VP ADD SELECTION** command adds the specified cells to the currently selected cells.

In *rangeObj*, pass a range object of cells to add to the current selection.

La celda activa no se modifica.

### Ejemplo

Actualmente tienes celdas seleccionadas:

	A	B	C	D	E	F	G
1							
2		Month	Area	Mr. Smith	Ms. Johnson	Ms. Williams	Mr. Jones
3		January	South	1898	1857	1651	1448
4			East	4859	4857	2548	4876
5			North	2458	1524	6150	4987
6			West	5787	1580	3975	4878
7		Total		15002	9818	14324	16189
8		February	South	6668	4374	17495	9999
9			East	5955	1677	7944	9400
10			North	1000	6722	2195	2777
11			West	6896	8355	7195	2058
12		Total		20519	21128	34829	24234
13		March	South	2577	2000	6185	2704
14			East	4859	4857	2548	4876
15			North	2458	1524	6150	4987
16			West	5787	1580	3975	4878
17		Total		15621	9961	18858	17115

The following code will add cells to your selection:

```
$currentSelection:=VP_Cells("myVPArea";3;4;2;3)
VP_ADD_SELECTION($currentSelection)
```

Resultado:

A	B	C	D	E	F	G
1						
2	Month	Area	Mr. Smith	Ms. Johnson	Ms. Williams	Mr. Jones
3	January	South	1898	1857	1651	1448
4		East	4859	4857	2548	4876
5		North	2458	1524	6150	4987
6		West	5787	1580	3975	4878
7	<b>Total</b>		15002	9818	14324	16189
8	February	South	6668	4374	17495	9999
9		East	5955	1677	7944	9400
10		North	1000	6722	2195	2777
11		West	6896	8355	7195	2058
12	<b>Total</b>		20519	21128	34829	24234
13	March	South	2577	2000	6185	2704
14		East	4859	4857	2548	4876
15		North	2458	1524	6150	4987
16		West	5787	1580	3975	4878
17	<b>Total</b>		15681	9961	19858	17415

Ver también

[VP Get active cell](#)

[VP Get selection](#)

[VP RESET SELECTION](#)

[VP SET ACTIVE CELL](#)

[VP SET SELECTION](#)

[VP SHOW CELL](#)

## VP ADD SHEET

`VP ADD SHEET ( vpAreaName : Text )`

`VP ADD SHEET ( vpAreaName : Text ; index : Integer )`

`VP ADD SHEET ( vpAreaName : Text ; index : Integer ; name : Text )`

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
index	Integer	->	Indice de la nueva hoja
name	Texto	->	Nombre de la hoja

### Descripción

The `VP ADD SHEET` command inserts a sheet in the document loaded in `vpAreaName`. In `vpAreaName`, pass the name of the 4D View Pro area.

En `index`, puede pasar un índice para la nueva hoja. In `sheet`, you can pass an index for the new sheet. If the passed `index` is inferior to or equal to 0, the command inserts the new sheet at the beginning.

La indexación comienza en 0.

In `name`, you can pass a name for the new sheet. The new name cannot contain the following characters: `*`, `:`, `[`, `]`, `? , \, /`

### Ejemplo

El documento tiene actualmente 3 hojas:



To insert a sheet at the third position (index 2) and name it "March":

```
VP ADD SHEET("ViewProArea";2;"March")
```



Ver también

[VP REMOVE SHEET](#)

## VP ADD SPAN

VP ADD SPAN ( *rangeObj* : Object )

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Objeto rango

### Descripción

The `VP ADD SPAN` command combines the cells in *rangeObj* as a single span of cells.

In *rangeObj*, pass a range object of cells. The cells in the range are joined to create a larger cell extending across multiple columns and/or rows. You can pass multiple cell ranges to create several spans at the same time. Note that if cell ranges overlap, only the first cell range is used.

- Sólo se muestran los datos de la celda superior izquierda. Data in the other combined cells is hidden until the span is removed.
- Hidden data in spanned cells is accessible via formulas (beginning with the upper-left cell).

### Ejemplo

To span the First quarter and Second quarter cells across the two cells beside them, and the South area cell across the two rows below it:

		First quart			Second qu		
		Jan	Feb	Mar	Apr	May	Jun
South area	John						
	Sahra						
	Dixie						

```
// First quarter range  
$q1:=VP Cells("ViewProArea";2;3;3;1)  
  
// Second quarter range  
$q2:=VP Cells("ViewProArea";5;3;3;1)  
  
// South area range  
$south:=VP Cells("ViewProArea";0;5;1;3)  
  
VP ADD SPAN(VP Combine ranges($q1;$q2;$south))
```

	First quarter			Second quarter		
	Jan	Feb	Mar	Apr	May	Jun
South area	John					
	Sahra					
	Dixie					

Ver también

[4D View Pro Range Object Properties](#)

[VP Get spans](#)

[VP REMOVE SPAN](#)

## VP ADD STYLESHEET

`VP ADD STYLESHEET ( vpAreaName : Text ; styleName : Text ; styleObj : Object { ; scope : Integer } )`

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
styleName	Texto	->	Nombre del estilo
styleObj	Objeto	->	Objeto definiendo las propiedades del atributo
scope	Integer	->	Alcance objetivo (por defecto = hoja actual)

### Descripción

The `VP ADD STYLESHEET` command creates or modifies the `styleName` style sheet based upon the combination of the properties specified in `styleObj` in the open document. If a style sheet with the same name and scope already exists in the document, this command will overwrite it with the new values.

Style sheets created by this command are saved with the document.

In `vpAreaName`, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

The `styleName` parameter lets you assign a name to the style sheet. If the name is already used within the same scope, the new style sheet replaces the existing one. Note that you can use the same name for different scopes (see below).

Within the `styleObj`, designate the settings for the style sheet (e.g., font, text decoration, alignment, borders, etc.). For the full list of style properties, see [Style object properties](#).

You can designate where to define the style sheet in the optional `scope` parameter using the sheet index (counting begins at 0) or with the following constants:

- `vk current sheet`
- `vk workbook`

If a `styleName` style sheet is defined at the workbook level and at a sheet level, the sheet level has priority over the workbook level when the style sheet is set.

To apply the style sheet, use the `VP SET DEFAULT STYLE` or `VP SET CELL STYLE` commands.

### Ejemplo

El código siguiente:

```

$styles:=New object
$styles.backColor:="green"

//Line Border Object
$borders:=New object("color";"green";"style";vk line style medium dash dot)

$styles.borderBottom:=$borders
$styles.borderLeft:=$borders
$styles.borderRight:=$borders
$styles.borderTop:=$borders

VP ADD STYLESHEET("ViewProArea";"GreenDashDotStyle";$styles)

//Para aplicar el estilo VP SET CELL STYLE(VP Cells("ViewProArea";1;1;2;2);New object("name";"GreenDashDo

```

will create and apply the following style object named *GreenDashDotStyle*:

```

{
  backColor:green,
  borderBottom:{color:green,style:10},
  borderLeft:{color:green,style:10},
  borderRight :{color:green,style:10},
  borderTop:{color:green,style:10}
}

```

## Ver también

[4D View Pro Style Objects and Style Sheets](#)

[VP Get stylesheet](#)

[VP Get stylesheets](#)

[VP REMOVE STYLESHEET](#)

[VP SET CELL STYLE](#)

[VP SET DEFAULT STYLE](#)

## VP All

**VP All ( vpAreaName : Text { ; sheet : Integer } ) : Object**

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
sheet	Integer	->	Índice de la hoja (hoja actual si se omite)
Resultado	Objeto	<-	Objeto rango de todas las celdas

### Descripción

The `VP ALL` command returns a new range object referencing all cells.

In `vpAreaName`, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

In the optional `sheet` parameter, you can designate a specific spreadsheet where the range will be defined (counting begins at 0). Si se omite o si se pasa `vk current sheet`, se utiliza la hoja de cálculo actual.

### Ejemplo

You want to define a range object for all of the cells of the current spreadsheet:

```
$all:=VP All("ViewProArea") // todas las celdas de la hoja actual
```

## Ver también

[VP Cell](#)  
[VP Cells](#)  
[VP Column](#)  
[VP Combine ranges](#)  
[VP Name](#)  
[VP Row](#)

## C

### VP Cell

VP Cell ( *vpAreaName* ; *column* : Integer ; *row* : Integer ; Text { ; *sheet* : Integer } ) : Object

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
<i>column</i>	Entero largo	->	Índice de la hoja (hoja actual si se omite)
<i>row</i>	Entero largo	->	Índice de la hoja (hoja actual si se omite)
<i>sheet</i>	Integer	->	Índice de la hoja (hoja actual si se omite)
Resultado	Objeto	<-	Objeto rango de todas las celdas

#### Descripción

The `VP Cell` command returns a new range object referencing a specific cell.

This command is intended for ranges of a single cell. To create a range object for multiple cells, use the [VP Cells](#) command.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

The *column* parameter defines the column of the cell range's position. Pase el índice de columna en este parámetro.

The *row* parameter defines the row of the cell range's position. Pase el índice de la línea en este parámetro.

In the optional *sheet* parameter, you can indicate the index of the sheet where the range will be defined. Si se omite o si se pasa `vk current sheet`, se utiliza la hoja de cálculo actual por defecto.

la indexación comienza en 0.

#### Ejemplo

You want to define a range object for the cell shown below (on the current spreadsheet):

1	A	B	C	D	E	F
2						
3						
4						
5						
6						
7						
8						
9						
10						

El código es el siguiente:

```
$cell:=VP Cell("ViewProArea";2;4) // C5
```

Ver también

[VP All](#)  
[VP Cells](#)  
[VP Column](#)  
[VP Combine ranges](#)  
[VP Name](#)  
[VP Row](#)

## VP Cells

**VP Cells ( vpAreaName : Text ; column: Integer ; row: Integer ; columnCount : Integer ; rowCount : Integer { ; sheet : Integer } ) : Object**

► Histórico

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
column	Integer	->	Índice de la columna
row	Integer	->	Índice de la línea
columnCount	Integer	->	Número de columnas
rowCount	Integer	->	Número de líneas
sheet	Integer	->	Índice de la hoja (hoja actual si se omite)
Resultado	Objeto	<-	Objeto rango de celdas

### Descripción

The `VP Cells` command returns a new range object referencing specific cells.

In `vpAreaName`, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

The `column` parameter defines the first column of the cell range. Pass the column index (counting begins at 0) in this parameter. If the range is within multiple columns, you should also use the `columnCount` parameter.

In the `row` parameter, you can define the row(s) of the cell range's position. Pass the row index (counting begins at 0) in this parameter. If the range is within multiple rows, you should also use the `rowCount` parameter.

The `columnCount` parameter allows you to define the total number of columns the range is within. `columnCount` must be greater than 0.

The `rowCount` parameter allows you to define the total number of rows the range is within. `rowCount` must be greater

than 0.

In the optional *sheet* parameter, you can designate a specific spreadsheet where the range will be defined (counting begins at 0). Si se omite o si se pasa `vk current sheet`, se utiliza la hoja de cálculo actual por defecto.

## Ejemplo

You want to define a range object for the following cells (on the current sheet):

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

El código es el siguiente:

```
$cells:=VP Cells("ViewProArea";2;4;2;3) // de C5 a D7
```

## Ver también

[VP All](#)  
[VP Cells](#)  
[VP Column](#)  
[VP Combine ranges](#)  
[VP Name](#)  
[VP Row](#)

## VP Column

`VP Column ( vpAreaName : Text ; column: Integer ; columnCount : Integer { ; sheet : Integer } ) : Object`

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
column	Integer	->	Índice de la columna
columnCount	Integer	->	Número de columnas
sheet	Integer	->	Índice de la hoja (hoja actual si se omite)
Resultado	Objeto	<-	Objeto rango de celdas

## Descripción

The `VP Column` command returns a new range object referencing a specific column or columns.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

The *column* parameter defines the first column of the column range. Pass the column index (counting begins at 0) in this parameter. If the range contains multiple columns, you should also use the optional *columnCount* parameter.

The optional *columnCount* parameter allows you to define the total number of columns of the range. *columnCount* must be greater than 0. If omitted, the value will be set to 1 by default and a column type range is created.

In the optional *sheet* parameter, you can designate a specific spreadsheet where the range will be defined (counting begins at 0). Si se omite o si se pasa `vk current sheet`, se utiliza la hoja de cálculo actual por defecto.

## Ejemplo

You want to define a range object for the column shown below (on the current spreadsheet):

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

El código es el siguiente:

```
$column:=VP Column("ViewProArea";3) // columna D
```

Ver también

[VP All](#)  
[VP Cells](#)  
[VP Column](#)  
[VP Combine ranges](#)  
[VP Name](#)  
[VP Row](#)  
[VP SET COLUMN ATTRIBUTES](#)

## VP COLUMN AUTOFIT

VP COLUMN AUTOFIT ( *rangeObj* : Object )

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Objeto rango

Descripción

The `VP COLUMN AUTOFIT` command automatically sizes the column(s) in *rangeObj* according to their contents.

In *rangeObj*, pass a range object containing a range of the columns whose size will be automatically handled.

Ejemplo

The following columns are all the same size and don't display some of the text:

	A	B	C	D	E
1	Hello Wor	The quick	TGIF		
2					
3					
4					
5					

Selecting the columns and running this code:

```
VP COLUMN AUTOFIT(VP Get selection("ViewProarea"))
```

... resizes the columns to fit the size of the contents:

	A	B	C	D	E
1	Hello World	The quick brown fox jumped over the lazy dog.	TGIF		
2					
3					
4					
5					

Ver también

[VP ROW AUTOFIT](#)

## VP Combine Ranges

VP Combine Ranges ( *rangeObj* : Object ; *otherRangeObj* : Object {...*otherRangeObjN* : Object } ) : Object

Parámetros	Tipo	Descripción
<i>rangeObj</i>	Objeto	-> Objeto rango
<i>otherRangeObj</i>	Objeto	-> Objeto rango
Resultado	Objeto	<- Objeto que contiene un rango combinado

### Descripción

The `VP Export to object` command returns the 4D View Pro object attached to the 4D View Pro area `vpAreaName`.

In *rangeObj*, pass the first range object.

In *otherRangeObj*, pass another range object(s) to combine with *rangeObj*.

The command incorporates *rangeObj* and *otherRangeObj* objects by reference.

### Ejemplo

You want to combine cell, column, and row range objects in a new, distinct range object:

```
$cell:=VP Cell("ViewProArea";2;4) // C5  
$column:=VP Column("ViewProArea";3) // column D  
$row:=VP Row("ViewProArea";9) // row 10  
  
$combine:=VP Combine ranges($cell;$column;$row)
```

### Ver también

[VP All](#)

[VP Cells](#)

[VP Column](#)

[VP Combine ranges](#)

[VP Name](#)

[VP Row](#)

[VP SET COLUMN ATTRIBUTES](#)

## VP Convert from 4D View

VP Convert from 4D View ( *4DViewDocument* : Blob ) : Object

Parámetros	Tipo	Descripción
<i>4DViewDocument</i>	Blob	-> Documento 4D View
Resultado	Objeto	<- Objeto 4D View Pro

### Descripción

The `VP Convert from 4D View` command allows you to convert a legacy 4D View document into a 4D View Pro object.

This command does not require that the legacy 4D View plug-in be installed in your environment.

In the *4DViewDocument* parameter, pass a BLOB variable or field containing the 4D View document to convert. The command returns a 4D View Pro object into which all the information originally stored within the 4D View document is converted to 4D View Pro attributes.

## Ejemplo

You want to get a 4D View Pro object from a 4D View area stored in a BLOB:

```
C_OBJECT($vpObj)
$vpObj:=VP Convert from 4D View($pvblob)
```

## VP Convert to picture

**VP Convert to picture ( *vpObject* : Object {; *rangeObj* : Object} ) : Picture**

Parámetros	Tipo		Descripción
<i>vpObject</i>	Objeto	->	Objeto 4D View Pro que contiene el área a convertir
<i>rangeObj</i>	Objeto	->	Objeto rango
Resultado	Objeto	<-	Imagen SVG del área

### Descripción

The `VP Convert to picture` command converts the *vpObject* 4D View Pro object (or the *rangeObj* range within *vpObject*) to a SVG picture.

Este comando es útil, por ejemplo:

- to embed a 4D View Pro document in an other document such as a 4D Write Pro document
- to print a 4D View Pro document without having to load it into a 4D View Pro area.

In *vpObject*, pass the 4D View Pro object that you want to convert. This object must have been previously parsed using [VP Export to object](#) or saved using [VP EXPORT DOCUMENT](#).

SVG conversion process requires that expressions and formats (cf. [Cell Format](#)) included in the 4D View Pro area be evaluated at least once, so that they can be correctly exported. If you convert a document that was not evaluated beforehand, expressions or formats may be rendered in an unexpected way.

In *rangeObj*, pass a range of cells to convert. By default, if this parameter is omitted, the whole document contents are converted.

Document contents are converted with respect to their viewing attributes, including formats (see note above), visibility of headers, columns and rows. The conversion of the following elements is supported:

- Text : style / font / size / alignment / orientation / rotation / format
- Cell background : color / image
- Bordes de las celdas : grosor / color / estilo
- Fusión de celdas
- Imágenes
- Altura de líneas
- Ancho de columnas
- Columnas / líneas ocultas.

Gridline visibility depends on document attribute defined with [VP SET PRINT INFO](#).

## Resultado

The command returns a picture in SVG format.

## Ejemplo

You want to convert a 4D View Pro area in SVG, preview the result, and send it to a picture variable:

```
C_OBJECT($vpAreaObj)
C_PICTURE($vPict)
$vpAreaObj:=VP Export to object("ViewProArea")
$vPict:=VP Convert to picture($vpAreaObj) //exportar toda el área
```

## Ver también

[VP EXPORT DOCUMENT](#)

[VP Export to object](#)

[VP SET PRINT INFO](#)

## VP Copy to object

► Histórico

`VP Copy to object ( rangeObj : Object {; options : Object} ) : Object`

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto rango
options	Objeto	->	Opciones adicionales
Resultado	Objeto	<-	Objeto devuelto. Contiene los datos copiados

### Descripción

The `VP Copy to object` command copies the contents, style and formulas from `rangeObj` to an object.

In `rangeObj`, pass the cell range with the values, formatting, and formulas to copy. If `rangeObj` is a combined range, only the first one is used.

You can pass an optional `options` parameter with the following properties:

Propiedad	Tipo	Descripción														
copy	Booleano	<code>True</code> (default) to keep the copied values, formatting and formulas after the command executes. <code>False</code> para eliminarlos.														
copyOptions	Entero largo	Especifica lo que se copia o mueve. Valores posibles: <table border="1"><thead><tr><th>Valor</th><th>Descripción</th></tr></thead><tbody><tr><td><code>vk clipboard options all</code> (por defecto)</td><td>Copies all data objects, including values, formatting, and formulas.</td></tr><tr><td><code>vk clipboard options formatting</code></td><td>Copia únicamente los formatos.</td></tr><tr><td><code>vk clipboard options formulas</code></td><td>Copia sólo las fórmulas.</td></tr><tr><td><code>vk clipboard options formulas and formatting</code></td><td>Copia las fórmulas y el formato.</td></tr><tr><td><code>vk clipboard options values</code></td><td>Copia sólo los valores.</td></tr><tr><td><code>vk clipboard options value and formatting</code></td><td>Copia los valores y el formato.</td></tr></tbody></table>	Valor	Descripción	<code>vk clipboard options all</code> (por defecto)	Copies all data objects, including values, formatting, and formulas.	<code>vk clipboard options formatting</code>	Copia únicamente los formatos.	<code>vk clipboard options formulas</code>	Copia sólo las fórmulas.	<code>vk clipboard options formulas and formatting</code>	Copia las fórmulas y el formato.	<code>vk clipboard options values</code>	Copia sólo los valores.	<code>vk clipboard options value and formatting</code>	Copia los valores y el formato.
Valor	Descripción															
<code>vk clipboard options all</code> (por defecto)	Copies all data objects, including values, formatting, and formulas.															
<code>vk clipboard options formatting</code>	Copia únicamente los formatos.															
<code>vk clipboard options formulas</code>	Copia sólo las fórmulas.															
<code>vk clipboard options formulas and formatting</code>	Copia las fórmulas y el formato.															
<code>vk clipboard options values</code>	Copia sólo los valores.															
<code>vk clipboard options value and formatting</code>	Copia los valores y el formato.															

The paste options defined in the [workbook options](#) are taken into account.

The command returns an object that contains the copied data.

## Ejemplo

This code sample first stores the contents, values, formatting and formulas from a range to an object, and then pastes them in another range:

```
var $originRange; $targetRange; $dataObject; $options : Object  
  
$originRange:=VP Cells("ViewProArea"; 0; 0; 2; 5)  
  
$options:=New object  
$options.copy:=True  
$options.copyOptions:=vk clipboard options all  
  
$dataObject:=VP Copy to object($originRange; $options)  
  
$targetRange:=VP Cell("ViewProArea"; 4; 0)  
VP PASTE FROM OBJECT($targetRange; $dataObject; vk clipboard options all)
```

Ver también

[VP PASTE FROM OBJECT](#)

[VP MOVE CELLS](#)

[VP Get workbook options](#)

[VP SET WORKBOOK OPTIONS](#)

## D

### VP DELETE COLUMNS

VP DELETE COLUMNS ( *rangeObj* : Object )

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto rango

#### Descripción

The `VP DELETE COLUMNS` command removes the columns in the *rangeObj*.

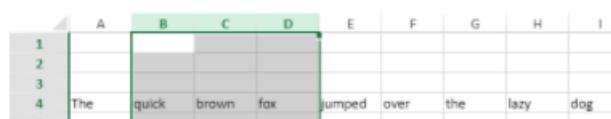
In *rangeObj*, pass an object containing a range of columns to remove. Si el rango pasado contiene:

- both columns and rows, only the columns are removed.
- únicamente las líneas, el comando no hace nada.

Columns are deleted from right to left.

## Ejemplo

To delete columns selected by the user (in the image below columns B, C, and D):



	A	B	C	D	E	F	G	H	I
1									
2									
3									
4	The	quick	brown	fox	jumped	over	the	lazy	dog
c									

utilice el siguiente código:

```
VP DELETE COLUMNS(VP Get selection("ViewProArea"))
```

Ver también

[VP All](#)  
[VP Cells](#)  
[VP Column](#)

## VP DELETE ROWS

VP DELETE ROWS ( *rangeObj* : Object )

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Objeto rango

Descripción

The `VP DELETE ROWS` command removes the rows in the *rangeObj*.

In *rangeObj*, pass an object containing a range of rows to remove. Si el rango pasado contiene:

- both columns and rows, only the rows are removed.
- sólo columnas, el comando no hace nada.

Las líneas se eliminan de abajo hacia arriba.

Ejemplo

To delete rows selected by the user (in the image below rows 1, 2, and 3):



A screenshot of a spreadsheet application showing a grid of text. The columns are labeled A through I at the top, and the rows are numbered 1 through 4 on the left. Row 1 contains "The". Row 2 contains "quick". Row 3 contains "brown". Row 4 contains "fox", "jumped", "over", "the", "lazy", and "dog". Rows 1, 2, and 3 are highlighted with a green selection bar, indicating they are selected for deletion.

utilice el siguiente código:

```
VP DELETE ROWS(VP Get selection("ViewProArea"))
```

Ver también

[VP All](#)  
[VP Cells](#)  
[VP Column](#)

E

## VP EXPORT DOCUMENT

VP EXPORT DOCUMENT ( *vpAreaName* : Text ; *filePath* : Text {; *paramObj* : Object} )

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
<i>filePath</i>	Texto	->	Ruta de acceso del documento
<i>paramObj</i>	Objeto	->	Opciones de exportación

## Descripción

En *vpAreaName*, pase el nombre del área 4D View Pro.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

In *filePath*, pass the destination path and name of the document to be exported. If you don't specify a path, the document will be saved at the same level as the Project folder.

You can specify the exported file's format by including an extension after the document's name:

- 4D View Pro ("4vp")
- Microsoft Excel ("xlsx")
- PDF ("pdf")
- CSV ("txt", o "csv")

If the extension is not included, but the format is specified in *paramObj*, the exported file will have the extension that corresponds to the format, except for the CSV format (no extension is added in this case).

The optional *paramObj* parameter allows you to define multiple properties for the exported 4D View Pro object, as well as launch a callback method when the export has completed.

Propiedad	Tipo	Descripción																		
format	texto	<p>(optional) When present, designates the exported file format: ".4vp" (default), ".csv", ".xlsx", or ".pdf". You can use the following constants:</p> <ul style="list-style-type: none"> <li>• <code>vk 4D View Pro format</code></li> <li>• <code>vk csv format</code></li> <li>• <code>vk MS Excel format</code></li> <li>• <code>vk pdf format</code></li> </ul> <p>4D adds the appropriate extension to the file name if needed. If the format specified doesn't correspond with the extension in <code>filePath</code>, it will be added to the end of <code>filePath</code>. If a format is not specified and no extension is provided in <code>filePath</code>, the default file format is used.</p>																		
contraseña	texto	Microsoft Excel only (optional) - Password used to protect the MS Excel document																		
formula	objeto	Callback method to be launched when the export has completed. Using a callback method is necessary when the export is asynchronous (which is the case for PDF and Excel formats) if you need some code to be executed after the export. The callback method must be used with the <code>Formula</code> command (see below for more information).																		
valuesOnly	booleano	Specifies that only the values from formulas (if any) will be exported.																		
includeFormatInfo	booleano	True to include formatting information, false otherwise (default is true). Formatting information is useful in some cases, e.g. for export to SVG. On the other hand, setting this property to false allows reducing export time.																		
sheetIndex	number	PDF only (optional) - Index of sheet to export (starting from 0). -2=all visible sheets (default), -1=current sheet only																		
pdfOptions	objeto	<p>PDF únicamente (opcional) - Opciones para la exportación en pdf</p> <table border="1"> <thead> <tr> <th>Propiedad</th><th>Tipo</th><th>Descripción</th></tr> </thead> <tbody> <tr> <td>creator</td><td>texto</td><td>name of the application that created the original document from which it was converted.</td></tr> <tr> <td>title</td><td>texto</td><td>título del documento.</td></tr> <tr> <td>author</td><td>texto</td><td>nombre de la persona que creó ese documento.</td></tr> <tr> <td>keywords</td><td>texto</td><td>palabras clave asociadas al documento.</td></tr> <tr> <td>subject</td><td>texto</td><td>asunto del documento.</td></tr> </tbody> </table>	Propiedad	Tipo	Descripción	creator	texto	name of the application that created the original document from which it was converted.	title	texto	título del documento.	author	texto	nombre de la persona que creó ese documento.	keywords	texto	palabras clave asociadas al documento.	subject	texto	asunto del documento.
Propiedad	Tipo	Descripción																		
creator	texto	name of the application that created the original document from which it was converted.																		
title	texto	título del documento.																		
author	texto	nombre de la persona que creó ese documento.																		
keywords	texto	palabras clave asociadas al documento.																		
subject	texto	asunto del documento.																		
csvOptions	objeto	<p>CSV únicamente (opcional) - Opciones para la exportación en csv</p> <table border="1"> <thead> <tr> <th>Propiedad</th><th>Tipo</th><th>Descripción</th></tr> </thead> <tbody> <tr> <td>range</td><td>objeto</td><td>Objeto rango de celdas</td></tr> <tr> <td>rowDelimiter</td><td>texto</td><td>Delimitador de línea. Por defecto: "¥r¥n"</td></tr> <tr> <td>columnDelimiter</td><td>texto</td><td>Delimitador de columna. Por defecto: ","</td></tr> </tbody> </table>	Propiedad	Tipo	Descripción	range	objeto	Objeto rango de celdas	rowDelimiter	texto	Delimitador de línea. Por defecto: "¥r¥n"	columnDelimiter	texto	Delimitador de columna. Por defecto: ","						
Propiedad	Tipo	Descripción																		
range	objeto	Objeto rango de celdas																		
rowDelimiter	texto	Delimitador de línea. Por defecto: "¥r¥n"																		
columnDelimiter	texto	Delimitador de columna. Por defecto: ","																		
<customProperty>	any	Any custom property that will be available through the \$3 parameter in the callback method.																		

Notes about Excel format:

- When exporting a 4D View Pro document into a Microsoft Excel-formatted file, some settings may be lost. For example, 4D methods and formulas are not supported by Excel. You can verify other settings with [this list from GrapeCity](#).
- Exporting in this format is run asynchronously, use the `formula` property of the `paramObj` for code to be executed after the export.

## Notas sobre formato PDF:

- When exporting a 4D View Pro document in PDF, the fonts used in the document are automatically embedded in the PDF file. Only OpenType fonts (.OTF or .TTF files) having a Unicode map can be embedded. If no valid font file is found for a font, a default font is used instead.
- Exporting in this format is run asynchronously, use the *formula* property of the *paramObj* for code to be executed after the export.

## Notas sobre el formato CSV :

- When exporting a 4D View Pro document to CSV, some settings may be lost, as only the text and values are saved.
- All the values are saved as double-quoted strings. For more information on delimiter-separated values, see [this article on Wikipedia](#).

Once the export operation is finished, `VP EXPORT DOCUMENT` automatically triggers the execution of the method set in the *formula* property of the *paramObj*, if used.

## Pasar un método retrollamada (fórmula)

When including the optional *paramObj* parameter, the `VP EXPORT DOCUMENT` command allows you to use the `Formula` command to call a 4D method which will be executed once the export has completed. The callback method will receive the following values in local variables:

Variable		Tipo	Descripción
\$1		texto	El nombre del objeto 4D View Pro
\$2		texto	Ruta de acceso del objeto 4D View Pro exportado
\$3		objeto	A reference to the command's <i>paramObj</i>
\$4		objeto	An object returned by the method with a status message
	.success	booleano	True si exporta con éxito, de lo contrario False.
	.errorCode	integer	Código de error. Puede ser devuelto por 4D o JavaScript.
	.errorMessage	texto	Mensaje de error. Puede ser devuelto por 4D o JavaScript.

## Ejemplo 1

You want to export the contents of the "VPArea" area to a 4D View Pro document on disk:

```
var $docPath: Text  
  
$docPath:="C:\\Bases\\ViewProDocs\\MyExport.4VP"  
VP EXPORT DOCUMENT("VPArea";$docPath)  
//MyExport.4VP is saved on your disk
```

## Ejemplo 2

You want to export the current sheet in PDF:

```
var $params: Object  
$params:=New object  
$params.format:=vk pdf format  
$params.sheetIndex:=-1  
$params.pdfOptions:=New object("title";"Annual Report";"author";Current user)  
VP EXPORT DOCUMENT("VPArea";"report.pdf";$params)
```

## Ejemplo 3

You want to export a 4D View Pro document in ".xlsx" format and call a method that will launch Microsoft Excel with the

document open once the export has completed:

```
$params:=New object
$params.formula:=Formula(AfterExport)
$params.format:=vp MS Excel format //".xlsx"
$params.valuesOnly:=True

VP EXPORT DOCUMENT("ViewProArea";"c:\\tmp\\convertedfile";$params)
```

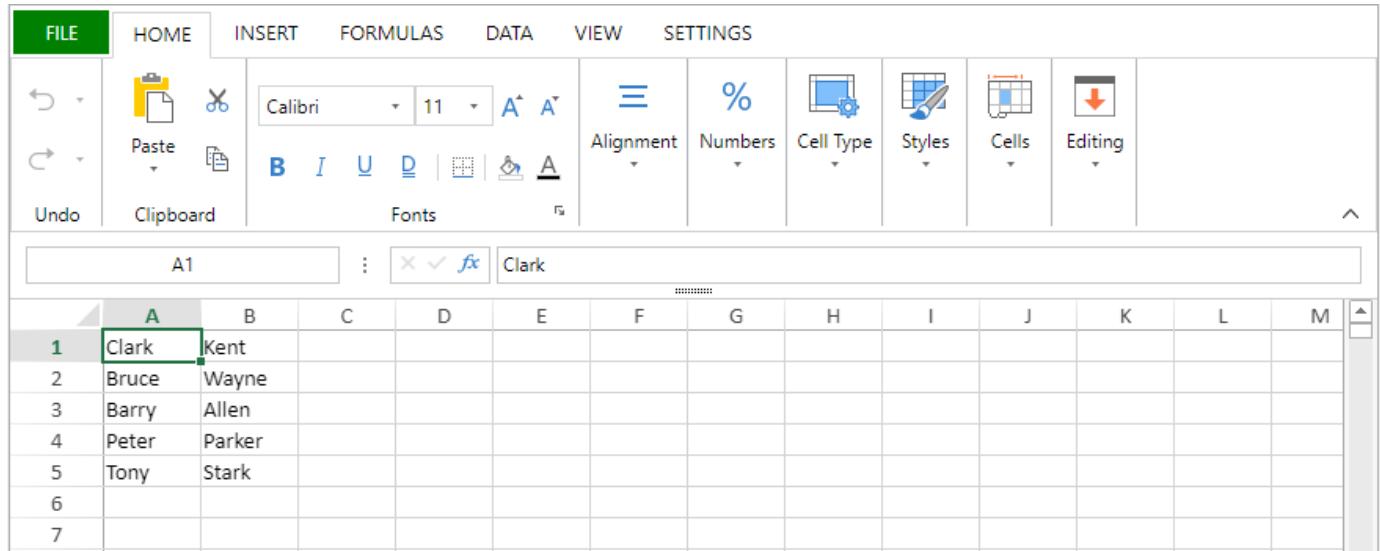
*AfterExport* method:

```
C_TEXT($1;$2)
C_OBJECT($3;$4)
$areaName:=$1
$filePath:=$2
$params:=$3
$status:=$4

If($status.success=False)
    ALERT($status.errorMessage)
Else
    LAUNCH EXTERNAL PROCESS("C:\\Program Files\\Microsoft Office\\Office15\\excel "+$filePath)
End if
```

#### Ejemplo 4

You want to export the current sheet to a `.txt` file with pipe-separated values:



The screenshot shows a Microsoft Excel spreadsheet with the following data in the first five rows:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Clark	Kent											
2	Bruce	Wayne											
3	Barry	Allen											
4	Peter	Parker											

```
var $params : Object
$params:=New object
$params.range:=VP Cells("ViewProArea";0;0;2;5)
$params.rowDelimiter:="\n"
$params.columnDelimiter:="|"
VP EXPORT DOCUMENT("ViewProArea";"c:\\tmp\\data.txt";New object("format";vk csv format;"csvOptions";$par
```

Aquí está el resultado:

```
"Clark"|"Kent"
"Bruce"|"Wayne"
"Barry"|"Allen"
"Peter"|"Parker"
"Tony"|"Stark"
```

## Ver también

[VP Convert to picture](#)  
[VP Export to object](#)  
[VP Column](#)  
[VP Print](#)

## VP Export to object

VP Export to object ( *vpAreaName* : Text {; *option* : Object} ) : Object

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
<i>option</i>	Objeto	->	Opción de exportación
Resultado	Objeto	<-	Objeto 4D View Pro

### Descripción

The `VP Export to object` command returns the 4D View Pro object attached to the 4D View Pro area *vpAreaName*.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

In the *option* parameter, you can pass the following export option, if required:

Propiedad	Tipo	Descripción
includeFormatInfo	booleano	True to include formatting information, false otherwise (default is true). Formatting information is useful in some cases, e.g. for export to SVG. On the other hand, setting this property to false allows reducing export time.

For more information on 4D View Pro objects, please refer to the [4D View Pro object](#) paragraph.

### Ejemplo 1

You want to get the "version" property of the current 4D View Pro area:

```
var $vpAreaObj : Object
var $vpVersion : Number
$vpAreaObj:=VP Export to object("vpArea")
// $vpVersion:=OB Get($vpAreaObj;"version")
$vpVersion:=$vpAreaObj.version
```

### Ejemplo 2

You want to export the area, excluding formatting information:

```
var $vpObj : Object
$vpObj:=VP Export to object("vpArea";New object("includeFormatInfo";False))
```

## Ver también

[VP Convert to picture](#)  
[VP EXPORT DOCUMENT](#)  
[VP IMPORT FROM OBJECT](#)

## VP Find

VP Find ( *rangeObj* : Object ; *searchValue* : Text ) : Object

VP Find ( *rangeObj* : Object ; *searchValue* : Text ; *searchCondition* : Object } ) : Object

VP Find ( *rangeObj* : Object ; *searchValue* : Text ; *searchCondition* : Object ; *replaceValue* : Text ) : Object

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Objeto rango
<i>searchValue</i>	Texto	->	Valor de búsqueda
<i>searchCondition</i>	Objeto	->	Objeto que contiene la(s) condición(es) de búsqueda
<i>replaceValue</i>	Texto	->	Valor de reemplazo
Resultado	Objeto	<-	Objeto rango

### Descripción

The `VP Find` command searches the *rangeObj* for the *searchValue*. Optional parameters can be used to refine the search and/or replace any results found.

In the *rangeObj* parameter, pass an object containing a range to search.

The *searchValue* parameter lets you pass the text to search for within the *rangeObj*.

You can pass the optional *searchCondition* parameter to specify how the search is performed. Se soportan las siguientes propiedades:

Propiedad	Tipo	Descripción									
afterColumn	Integer	The number of the column just before the starting column of the search. If the <i>rangeObj</i> is a combined range, the column number given must be from the first range. Default value: -1 (beginning of the <i>rangeObj</i> )									
afterRow	Integer	The number of the row just before the starting row of the search. If the <i>rangeObj</i> is a combined range, the row number given must be from the first range. Default value: -1 (beginning of the <i>rangeObj</i> )									
all	Booleano	<ul style="list-style-type: none"> <li>True - All cells in <i>rangeObj</i> corresponding to <i>searchValue</i> are returned</li> <li>False - (default value) Only the first cell in <i>rangeObj</i> corresponding to <i>searchValue</i> is returned</li> </ul>									
flags	Integer	<table border="1"> <tr> <td><code>vk find flag exact match</code></td><td>The entire content of the cell must completely match the search value</td></tr> <tr> <td><code>vk find flag ignore case</code></td><td>Capital and lower-case letters are considered the same. Ej.: "a" es considerada como idéntica a "A".</td></tr> <tr> <td><code>vk find flag none</code></td><td>no se consideran los indicadores de búsqueda (por defecto)</td></tr> <tr> <td><code>vk find flag use wild cards</code></td><td>           Wildcard characters (*,?) can be used in the search string. Wildcard characters can be used in any string comparison to match any number of characters:           <ul style="list-style-type: none"> <li>* for zero or multiple characters (for example, searching for "bl*" can find "bl", "black", or "blob")</li> <li>? ? ? ? for a single character (for example, searching for "h?t" can find "hot", or "hit")</li> </ul> </td></tr> </table>	<code>vk find flag exact match</code>	The entire content of the cell must completely match the search value	<code>vk find flag ignore case</code>	Capital and lower-case letters are considered the same. Ej.: "a" es considerada como idéntica a "A".	<code>vk find flag none</code>	no se consideran los indicadores de búsqueda (por defecto)	<code>vk find flag use wild cards</code>	Wildcard characters (*,?) can be used in the search string. Wildcard characters can be used in any string comparison to match any number of characters: <ul style="list-style-type: none"> <li>* for zero or multiple characters (for example, searching for "bl*" can find "bl", "black", or "blob")</li> <li>? ? ? ? for a single character (for example, searching for "h?t" can find "hot", or "hit")</li> </ul>	
<code>vk find flag exact match</code>	The entire content of the cell must completely match the search value										
<code>vk find flag ignore case</code>	Capital and lower-case letters are considered the same. Ej.: "a" es considerada como idéntica a "A".										
<code>vk find flag none</code>	no se consideran los indicadores de búsqueda (por defecto)										
<code>vk find flag use wild cards</code>	Wildcard characters (*,?) can be used in the search string. Wildcard characters can be used in any string comparison to match any number of characters: <ul style="list-style-type: none"> <li>* for zero or multiple characters (for example, searching for "bl*" can find "bl", "black", or "blob")</li> <li>? ? ? ? for a single character (for example, searching for "h?t" can find "hot", or "hit")</li> </ul>										
Estos marcadores se pueden combinar. Por ejemplo:											
<code>\$search.flags:=vk find flag use wild cards+vk find flag ignore case</code>											
order	Integer	<table border="1"> <tr> <td><code>vk find order by columns</code></td><td>La búsqueda se realiza por columnas. Each row of a column is searched before the search continues to the next column.</td></tr> <tr> <td><code>vk find order by rows</code></td><td>La búsqueda se realiza por líneas. Each column of a row is searched before the search continues to the next row (default)</td></tr> </table>	<code>vk find order by columns</code>	La búsqueda se realiza por columnas. Each row of a column is searched before the search continues to the next column.	<code>vk find order by rows</code>	La búsqueda se realiza por líneas. Each column of a row is searched before the search continues to the next row (default)					
<code>vk find order by columns</code>	La búsqueda se realiza por columnas. Each row of a column is searched before the search continues to the next column.										
<code>vk find order by rows</code>	La búsqueda se realiza por líneas. Each column of a row is searched before the search continues to the next row (default)										
target	Integer	<table border="1"> <tr> <td><code>vk find target formula</code></td><td>La búsqueda se realiza en la fórmula de la celda</td></tr> <tr> <td><code>vk find target tag</code></td><td>La búsqueda se realiza en la etiqueta de la celda</td></tr> <tr> <td><code>vk find target text</code></td><td>The search is performed in the cell text (default)</td></tr> </table>	<code>vk find target formula</code>	La búsqueda se realiza en la fórmula de la celda	<code>vk find target tag</code>	La búsqueda se realiza en la etiqueta de la celda	<code>vk find target text</code>	The search is performed in the cell text (default)			
<code>vk find target formula</code>	La búsqueda se realiza en la fórmula de la celda										
<code>vk find target tag</code>	La búsqueda se realiza en la etiqueta de la celda										
<code>vk find target text</code>	The search is performed in the cell text (default)										
Estos marcadores se pueden combinar. Por ejemplo:											
<code>\$search.target:=vk find target formula+vk find target text</code>											

In the optional *replaceValue* parameter, you can pass text to take the place of any instance of the text in *searchValue* found in the *rangeObj*.

## Objeto devuelto

The function returns a range object describing each search value that was found or replaced. An empty range object is

returned if no results are found.

## Ejemplo 1

To find the first cell containing the word "Total":

```
var $range;$result : Object  
  
$range:=VP All("ViewProArea")  
  
$result:=VP Find($range;"Total")
```

## Ejemplo 2

To find "Total" and replace it with "Grand Total":

```
var $range;$condition;$result : Object  
  
$range:=VP All("ViewProArea")  
  
$condition:=New object  
$condition.target:=vk find target text  
$condition.all:=True //Search entire document  
$condition.flags:=vk find flag exact match  
  
// Reemplazar las celdas que contienen sólo "Total" en la hoja actual con "Grand Total"  
$result:=VP Find($range;"Total";$condition;"Grand Total")  
  
// Comprobar si el objeto de rango está vacío  
If($result.ranges.length=0)  
    ALERT("No result found")  
Else  
    ALERT($result.ranges.length+" results found")  
End if
```

## VP FLUSH COMMANDS

VP FLUSH COMMANDS ( *vpAreaName* : Text )

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro

### Descripción

The `VP FLUSH COMMANDS` command immediately executes stored commands and clears the command buffer.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

In order to increase performance and reduce the number of requests sent, the 4D View Pro commands called by the developer are stored in a command buffer. When called, `VP FLUSH COMMANDS` executes the commands as a batch when leaving the method and empties the contents of the command buffer.

## Ejemplo

You want to trace the execution of the commands and empty the command buffer:

```

VP SET TEXT VALUE(VP Cell("ViewProArea1";10;1);"INVOICE")
VP SET TEXT VALUE(VP Cell("ViewProArea1";10;2);"Invoice date: ")
VP SET TEXT VALUE(VP Cell("ViewProArea1";10;3);"Due date: ")

VP FLUSH COMMANDS(("ViewProArea1")
TRACE

```

## VP Font to object

VP Font to object ( *font* : Text ) : Object

Parámetros	Tipo		Descripción
font	Texto	->	Cadena abreviada para la fuente

### Descripción

The `VP Font to object` utility command returns an object from a font shorthand string.

In the *font* parameter, pass a font shorthand string to specify the different properties of a font (e.g., "12 pt Arial"). You can learn more about font shorthand strings [in this page](#) for example.

The returned object contains defined font attributes as properties. For more information about the available properties, see the [VP Object to font](#) command.

### Ejemplo 1

Este código:

```
$font:=VP Font to object("16pt arial")
```

devolverá el objeto \$font:

```
{
  family:arial
  size:16pt
}
```

### Ejemplo 2

See example for [VP Object to font](#).

Ver también

[4D View Pro Style Objects and Style Sheets](#)  
[VP Object to font](#)  
[VP SET CELL STYLE](#)  
[VP SET DEFAULT STYLE](#)

## G

## VP Get active cell

VP Get active cell ( *vpAreaName* : Text { ; *sheet* : Integer } ) : Object

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
sheet	Integer	->	Índice de la hoja (hoja actual si se omite)
Resultado	Objeto	<-	Objeto rango de una sola celda

## Descripción

The `VP Get active cell` command returns a new range object referencing the cell which has the focus and where new data will be entered (the active cell).

In `vpAreaName`, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

In the optional `sheet` parameter, you can designate a specific spreadsheet where the range will be defined (counting begins at 0). Si se omite o si se pasa `vk current sheet`, se utiliza la hoja de cálculo actual.

## Ejemplo

A	B	C	D	E	F	G
1						
2	Month	Area	Mr. Smith	Ms. Johnson	Ms. Williams	Mr. Jones
3	January	South	1898	1857	1651	1448
4		East	4859	4857	2548	4876
5		North	2458	1524	6150	4987
6		West	5787	1580	3975	4878
7	<b>Total</b>		15002	9818	14324	16189
8	February	South	6668	4374	17495	9999
9		East	5955	1677	7944	9400
10		North	1000	6722	2195	2777
11		West	6896	8355	7195	2058
12	<b>Total</b>		20519	21128	34829	24234
13	March	South	2577	2000	6185	2704
14		East	4859	4857	2548	4876
15		North	2458	1524	6150	4987
16		West	5787	1580	3975	4878
17	<b>Total</b>		15681	9961	18858	17445

The following code will retrieve the coordinates of the active cell:

```
$activeCell:=VP Get active cell("myVPArea")

//returns a range object containing:
//$activeCell.ranges[0].column=3
//$activeCell.ranges[0].row=4
//$activeCell.ranges[0].sheet=0
```

## Ver también

[VP ADD SELECTION](#)  
[VP Get selection](#)  
[VP RESET SELECTION](#)  
[VP SET ACTIVE CELL](#)  
[VP SET SELECTION](#)  
[VP SHOW CELL](#)

## VP Get cell style

`VP Get cell style ( rangeObj : Object ) : Object`

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto rango
Resultado	Objeto	<-	Objeto style

## Descripción

The `VP Get cell style` command returns a [style object](#) for the first cell in the *rangeObj*.

In *rangeObj*, pass a range containing the style to retrieve.

- If *rangeObj* contains a cell range, the cell style is returned.
- If *rangeObj* contains a range that is not a cell range, the style of the first cell in the range is returned.
- If *rangeObj* contains several ranges, only the style of the first cell in the first range is returned.

## Ejemplo

To get the details about the style in the selected cell (B2):

A	B	C
1		
2	Hello World	
3		

Este código:

```
$cellStyle:=VP Get cell style(VP Get selection("myDoc"))
```

... devolverá este objeto:

```
{
  "backColor": "Azure",
  "borderBottom": {
    "color": "#800080",
    "style": 5
  },
  "font": "8pt Arial",
  "foreColor": "red",
  "hAlign": 1,
  "isVerticalText": "true",
  "vAlign": 0
}
```

## Ver también

[VP GET DEFAULT STYLE](#)

[VP SET CELL STYLE](#)

## VP Get column attributes

`VP Get column attributes ( rangeObj : Object ) : Collection`

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto rango
Resultado	Collection	<-	Colección de propiedades de columnas

## Descripción

The `VP Get column attributes` command returns a collection of properties for any column in the `rangeObj`.

In `rangeObj`, pass an object containing a range of the columns whose attributes will be retrieved.

The returned collection contains any properties for the columns, whether or not they have been set by the [VP SET COLUMN ATTRIBUTES](#) command.

## Ejemplo

El código siguiente:

```
C_OBJECT($range)
C_COLLECTION($attr)

$range:=VP Column("ViewProArea";1;2)
$attr:=VP Get column attributes($range)
```

... will return a collection of the attributes within the given range:

length	2
\$attr[0]	{"width":150,"pageBreak":false,"visible":true,"resizable":false,"header":"Hello World"}
\$attr[0].header	"Hello World"
\$attr[0].pageBreak	False
\$attr[0].resizable	False
\$attr[0].visible	True
\$attr[0].width	150
\$attr[1]	{"width":62,"pageBreak":false,"visible":true,"resizable":true,"header":"C"}
\$attr[1].header	"C"
\$attr[1].pageBreak	False
\$attr[1].resizable	True
\$attr[1].visible	True
\$attr[1].width	62

## Ver también

[VP Get row attributes](#)  
[VP SET COLUMN ATTRIBUTES](#)  
[VP SET ROW ATTRIBUTES](#)

## VP Get column count

`VP Get column count ( vpAreaName : Text { ; sheet : Integer } ) : Integer`

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre del área 4D View Pro en el formulario
sheet	Integer	->	Índice de la hoja (hoja actual si se omite)
Resultado	Integer	<-	Número total de columnas

## Descripción

The `VP Get column count` command returns the total number of columns from the designated `sheet`.

In *vpAreaName*, pass the name property of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

You can define where to get the column count in the optional *sheet* parameter using the sheet index (counting begins at 0). Si se omite o si se pasa `vk current sheet`, se utiliza la hoja de cálculo actual.

## Ejemplo

The following code returns the number of columns in the 4D View Pro area:

```
C_Integer($colCount)
$colCount:=VP Get column count("ViewProarea")
```

## Ver también

[VP Get row count](#)  
[VP SET COLUMN COUNT](#)  
[VP SET ROW COUNT](#)

## VP Get current sheet

**VP Get current sheet ( *vpAreaName* : Text )**

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
Resultado	Integer	<-	Indice de la hoja actual

## Descripción

The `VP Get current sheet` command returns the index of the current sheet in *vpAreaName*. The current sheet is the selected sheet in the document.

In *vpAreaName*, pass the name of the 4D View Pro area.

La indexación comienza en 0.

## Ejemplo

Cuando se selecciona la tercera hoja:



El comando devuelve 2:

```
$index:=VP Get current sheet("ViewProArea")
```

## Ver también

[VP SET CURRENT SHEET](#)

## VP Get default style

**VP Get default style ( *vpAreaName* : Text { ; *sheet* : Integer } ) : Integer**

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre del área 4D View Pro en el formulario
sheet	Integer	->	Índice de la hoja (hoja actual si se omite)
Resultado	Integer	<-	Número total de columnas

## Descripción

The `VP Get default style` command returns a default style object for a sheet. The returned object contains basic document rendering properties as well as the default style settings (if any) previously set by the [VP SET DEFAULT STYLE](#) method. For more information about style properties, see [Style Objects & Style Sheets](#).

In `vpAreaName`, pass the name property of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

You can define where to get the column count in the optional `sheet` parameter using the sheet index (counting begins at 0). Si se omite o si se pasa `vk current sheet`, se utiliza la hoja de cálculo actual.

## Ejemplo

To get the details about the default style for this document:

	A	B	C
1			
2	Hello World!		
3			
4			
5			
6			
7			
8			

Este código:

```
$defaultStyle:=VP Get default style("myDoc")
```

will return this information in the `$defaultStyle` object:

```
{
  backColor:#E6E6FA,
  hAlign:0,
  vAlign:0,
  font:12pt papyrus
}
```

## Ver también

[VP Get cell style](#)  
[VP SET DEFAULT STYLE](#)

## VP Get formula

`VP Get formula ( rangeObj : Object ) : Text`

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto rango
Resultado	Texto	<-	Formula

## Descripción

The `VP Get formula` command retrieves the formula from a designated cell range.

In `rangeObj`, pass a range whose formula you want to retrieve. If `rangeObj` designates multiple cells or multiple ranges, the formula of the first cell is returned. If `rangeObj` is a cell that does not contain a formula, the method returns an empty string.

## Ejemplo

```
//definir una fórmula
VP SET FORMULA(VP Cell("ViewProArea";5;2);"SUM($A$1:$C$10)")

$result:=VP Get formula(VP Cell("ViewProArea";5;2)) // $result="SUM($A$1:$C$10)"
```

## Ver también

[VP Get formulas](#)  
[VP SET FORMULA](#)  
[VP SET ROW COUNT](#)

## VP Get formula by name

`VP Get formula by name ( vpAreaName : Text ; name : Text { ; scope : Number } ) : Object`

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
name	Texto	->	Nombre del rango nombrado
scope	Número	->	Alcance objetivo (por defecto=hoja actual)
Resultado	Texto	<-	Definición de la fórmula o rango con nombre

## Descripción

The `VP Get formula by name` command returns the formula and comment corresponding to the named range or named formula passed in the `name` parameter, or null if it does not exist in the defined scope.

In `vpAreaName`, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

Pass the named range or named formula that you want to get in `name`. Note that named ranges are returned as formulas containing absolute cell references.

You can define where to get the formula in `scope` using either the sheet index (counting begins at 0) or the following constants:

- `vk current sheet`
- `vk workbook`

## Objeto devuelto

El objeto devuelto contiene las siguientes propiedades:

Propiedad	Tipo	Descripción
formula	Texto	Text of the formula corresponding to the named formula or named range. For named ranges, the formula is a sequence of absolute coordinates.
comment	Texto	Comment corresponding to the named formula or named range

## Ejemplo

```
$range:=VP Cell("ViewProArea";0;0)
VP ADD RANGE NAME("Total1";$range)

$formula:=VP Get formula by name("ViewProArea";"Total1")
//$formula.formula=Sheet1!$A$1

$formula:=VP Get formula by name("ViewProArea";"Total")
//$formula=null (if not existing)
```

## Ver también

[VP ADD FORMULA NAME](#)

[VP ADD RANGE NAME](#)

[VP Get names](#)

## VP Get formulas

VP Get formulas ( *rangeObj* : Object ) : Collection

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto rango
Resultado	Collection	<-	Colección de valores de una fórmula

## Descripción

The `VP Get formulas` command retrieves the formulas from a designated *rangeObj*.

In *rangeObj*, pass a range whose formulas you want to retrieve. If *rangeObj* designates multiple ranges, the formula of the first range is returned. If *rangeObj* does not contain any formulas, the command returns an empty string.

La colección devuelta es bidimensional:

- The first-level collection contains subcollections of formulas. Cada subcolección representa una línea.
- Each subcollection defines cell values for the row. Values are text elements containing the cell formulas.

## Ejemplo

You want to retrieve the formulas in the Sum and Average columns from this document:

	A	B	C	D	E	F	G
1		Data					
2		1	2	3		6	2
3		4	5	6		15	5
4		7	8	9		24	8.5
5							

Puede utilizar este código:

```

$`formulas:=VP Get formulas(VP Cells("ViewProArea";5;1;2;3))
//`formulas[0]=[Sum(B2:D2),Average(B2:D2)]
//`formulas[1]=[Sum(B3:D3),Average(B3:D3)]
//`formulas[2]=[Sum(B4:D4),Average(C4:D4)]
```

Ver también

[VP Get formula](#)  
[VP Get values](#)  
[VP SET FORMULAS](#)  
[VP SET VALUES](#)

## VP Get frozen panes

VP Get frozen panes ( vpAreaName : Text { ; sheet : Integer } ) : Object

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
sheet	Integer	->	Índice de la hoja (hoja actual si se omite)
Resultado	Objeto	<-	Object containing frozen column and row information

### Descripción

The `VP Get frozen panes` command returns an object with information about the frozen columns and rows in `vpAreaName`.

In `vpAreaName`, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

In the optional `sheet` parameter, you can designate a specific spreadsheet where the range will be defined (counting begins at 0). Si se omite o si se pasa `vk current sheet`, se utiliza la hoja de cálculo actual.

### Objeto devuelto

The command returns an object describing the frozen columns and rows. Este objeto puede contener las siguientes propiedades:

Propiedad	Tipo	Descripción
columnCount	Integer	The number of frozen columns on the left of the sheet
trailingColumnCount	Integer	The number of frozen columns on the right of the sheet
rowCount	Integer	El número de líneas congeladas en la parte superior de la hoja
trailingRowCount	Integer	The number of frozen rows on the bottom of the sheet

### Ejemplo

You want to retrieve information about the number of frozen columns and rows:

```

var $panesObj : Object
$panesObj:=VP Get frozen panes("ViewProArea")
```

El objeto devuelto contiene, por ejemplo:

Expression	Value
✓ \$paneObj	{"columnCount":3,"trailingColumnCount":0,"rowCount":1,"trailingRowCount":0}
└ columnCount	3
└ rowCount	1
└ trailingColumnCount	0
└ trailingRowCount	0

Ver también

[VP SET FROZEN PANES](#)

## VP Get names

VP Get names ( vpAreaName : Text { ; scope : Number } ) : Collection

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
scope	Número	->	Alcance objetivo (por defecto= hoja actual)
Resultado	Collection	<-	Nombres existentes en el alcance definido

### Descripción

The `VP Get names` command returns a collection of all defined "names" in the current sheet or in the scope designated by the `scope` parameter.

In `vpAreaName`, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

You can define where to get the names in `scope` using either the sheet index (counting begins at 0) or the following constants:

- `vk current sheet`
- `vk workbook`

### Colección devuelta

The returned collection contains one object per name. Las propiedades de objetos siguientes pueden ser devueltas:

Propiedad	Tipo	Descripción
<code>result[ ].name</code>	Texto	nombre de celda o de rango
<code>result[ ].formula</code>	Texto	formula
<code>result[ ].comment</code>	Texto	Comentario asociado al nombre

Available properties depend on the type of the named element (named cell, named range, or named formula).

### Ejemplo

```
var $list : Collection
$list:=VP Get names("ViewProArea";2) //names in 3rd sheet
```

Ver también

[VP ADD FORMULA NAME](#)

[VP ADD RANGE NAME](#)

[VP Get formula by name](#)

[VP Name](#)

## VP Get print info

VP Get print info ( vpAreaName : Text { ; sheet : Integer } ) : Object

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
sheet	Integer	->	Índice de la hoja (hoja actual si se omite)
Resultado	Objeto	<-	Objeto que contiene la información de impresión

### Descripción

The `VP Get print info` command returns an object containing the print attributes of the `vpAreaName`.

Pass the name of the 4D View Pro area in `vpAreaName`. If you pass a name that does not exist, an error is returned.

In the optional `sheet` parameter, you can designate a specific spreadsheet (counting begins at 0) whose printing attributes you want returned. Si se omite o si se pasa `vk current sheet`, se utiliza la hoja de cálculo actual.

### Ejemplo

Este código:

```
$pinfo:=VP Get print info("ViewProArea")
```

... returns the print attributes of the 4D View Pro area set in the `VP SET PRINT INFO` command:

```
{
bestFitColumns:false,
bestFitRows:false,
blackAndWhite:false,
centering:0,
columnEnd:8,
columnStart:0,
firstPageNumber:1,
fitPagesTall:1,
fitPagesWide:1,
footerCenter:"&BS.H.I.E.L.D. &A Sales Per Region",
footerCenterImage:, 
footerLeft:, 
footerLeftImage:, 
footerRight:"page &P of &N", 
footerRightImage:, 
headerCenter:, 
headerCenterImage:, 
headerLeft:&G, 
headerLeftImage:logo.png, 
headerRight:, 
headerRightImage:, 
margin:{top:75,bottom:75,left:70,right:70,header:30,footer:30}, 
orientation:2, 
pageOrder:0, 
pageRange:, 
paperSize:{width:850,height:1100,kind:1}, 
qualityFactor:2, 
repeatColumnEnd:-1, 
repeatColumnStart:-1, 
repeatRowEnd:-1, 
repeatRowStart:-1, 
rowEnd:24, 
rowStart:0, 
showBorder:false, 
showColumnHeader:0, 
showGridLine:false, 
showRowHeader:0, 
useMax:true, 
watermark:[], 
zoomFactor:1
}
```

Ver también

[4D View Pro Print Attributes](#)  
[VP SET PRINT INFO](#)

## VP Get row attributes

VP Get row attributes ( rangeObj : Object ) : Collection

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto rango
Resultado	Collection	<-	Colección de propiedades de la línea

### Descripción

The `VP Get row attributes` command returns a collection of properties for any row in the `rangeObj`.

In `rangeObj`, pass an object containing a range of the rows whose attributes will be retrieved.

The returned collection contains any properties for the rows, whether or not they have been set by the [VP SET ROW ATTRIBUTES](#) method.

## Ejemplo

The following code returns a collection of the attributes within the given range:

```
var $range : Object
var $attr : Collection

$range:=VP Column("ViewProArea";1;2)
$attr:=VP Get row attributes($range)
```

length	2
\$attr[0]	{"height":75,"pageBreak":false,"visible":true,"resizable":true,"header":"june"}
\$header	"june"
\$height	75
\$pageBreak	False
\$resizable	True
\$visible	True
\$attr[1]	{"height":20,"pageBreak":false,"visible":true,"resizable":true,"header":"3"}
\$header	"3"
\$height	20
\$pageBreak	False
\$resizable	True
\$visible	True

## Ver también

[VP Get column attributes](#)  
[VP SET COLUMN ATTRIBUTES](#)  
[VP SET ROW ATTRIBUTES](#)

## VP Get row count

`VP Get row count ( vpAreaName : Text {; sheet : Integer } ) : Integer`

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre del área 4D View Pro en el formulario
sheet	Integer	->	Índice de la hoja (hoja actual si se omite)
Resultado	Integer	<-	Número total de líneas

## Descripción

The `VP Get row count` command returns the total number of rows from the designated `sheet`.

In `vpAreaName`, pass the name property of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

You can define where to get the row count in the optional `sheet` parameter using the sheet index (counting begins at 0). Si se omite o si se pasa `vk current sheet`, se utiliza la hoja de cálculo actual.

## Ejemplo

The following code returns the number of rows in the 4D View Pro area:

```

var $rowCount : Integer
$rowCount:=VP Get row count("ViewProarea")

```

Ver también

[VP Get column count](#)  
[VP SET COLUMN COUNT](#)  
[VP SET ROW COUNT](#)

## VP Get selection

VP Get selection ( *vpAreaName* : Text {; *sheet* : Integer } ) : Object

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre del área 4D View Pro en el formulario
<i>sheet</i>	Integer	->	Índice de la hoja (hoja actual si se omite)
Resultado	Objeto	<-	Objeto rango de celdas

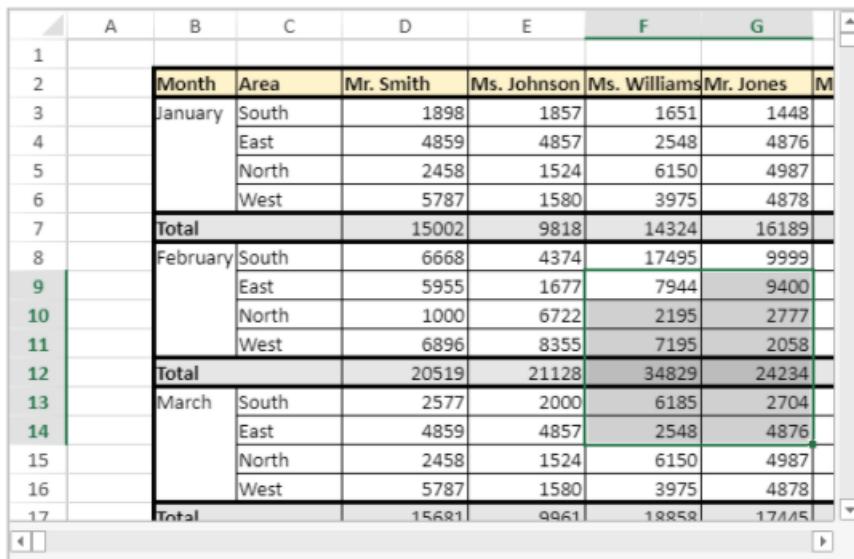
### Descripción

The `VP Get selection` command returns a new range object referencing the current selected cells.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

In the optional *sheet* parameter, you can designate a specific spreadsheet where the range will be defined (counting begins at 0). Si se omite o si se pasa `vk current sheet`, se utiliza la hoja de cálculo actual.

### Ejemplo



	A	B	C	D	E	F	G
1							
2		Month	Area	Mr. Smith	Ms. Johnson	Ms. Williams	Mr. Jones
3	January	South		1898	1857	1651	1448
4		East		4859	4857	2548	4876
5		North		2458	1524	6150	4987
6		West		5787	1580	3975	4878
7	<b>Total</b>			15002	9818	14324	16189
8	February	South		6668	4374	17495	9999
9		East		5955	1677	7944	9400
10		North		1000	6722	2195	2777
11		West		6896	8355	7195	2058
12	<b>Total</b>			20519	21128	34829	24234
13	March	South		2577	2000	6185	2704
14		East		4859	4857	2548	4876
15		North		2458	1524	6150	4987
16		West		5787	1580	3975	4878
17	<b>Total</b>			15621	99611	182521	171151

The following code will retrieve the coordinates of all the cells in the current selection:

```

$currentSelection:=VP Get selection("myVPArea")

//devuelve un objeto rango que contiene:
//$currentSelection.ranges[0].column=5
//$currentSelection.ranges[0].columnCount=2
//$currentSelection.ranges[0].row=8
//$currentSelection.ranges[0].rowCount=6

```

Ver también

[VP ADD SELECTION](#)  
[VP Get active cell](#)  
[VP SET ACTIVE CELL](#)  
[VP SET SELECTION](#)  
[VP SHOW CELL](#)

## VP Get sheet count

VP Get sheet count ( *vpAreaName* : Text ) : Integer

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
Resultado	Integer	<-	Número de hojas

### Descripción

The `VP Get sheet count` command returns the number of sheets in the document loaded in *vpAreaName*.

In *vpAreaName*, pass the name of the 4D View Pro area.

### Ejemplo

En el siguiente documento:



Get the sheet count and set the current sheet to the last sheet:

```
$count:=VP Get sheet count("ViewProArea")
//set the current sheet to the last sheet (indexing starts at 0)
VP SET CURRENT SHEET("ViewProArea";$count-1)
```



Ver también

[VP Get sheet index](#)  
[VP SET SHEET COUNT](#)

## VP Get sheet index

VP Get sheet index ( *vpAreaName* : Text ; *name* : Text ) : Integer

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
<i>name</i>	Texto	->	Nombre de la hoja
Resultado	Integer	<-	Índice de la hoja

### Descripción

The `VP Get sheet index` command returns the index of a sheet based on its name in *vpAreaName*.

In *vpAreaName*, pass the name of the 4D View Pro area.

In *name*, pass the name of the sheet whose index will be returned. If no sheet named *name* is found in the document, the method returns -1.

La indexación comienza en 0.

## Ejemplo

En el siguiente documento:



Get the index of the sheet called "Total first quarter":

```
$index:=VP Get sheet index("ViewProArea";"Total premier trimestre") //devuelve 2
```

## Ver también

[VP Get sheet count](#)

[VP Get sheet name](#)

## VP Get sheet name

VP Get sheet name ( *vpAreaName* : Text ; *sheet* : Integer ) : Text

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
<i>sheet</i>	Integer	->	Índice de la hoja
Resultado	Texto	<-	Nombre de la hoja

## Descripción

The `VP Get sheet name` command returns the name of a sheet based on its index in *vpAreaName*.

In *vpAreaName*, pass the name of the 4D View Pro area.

In *sheet*, pass the index of the sheet whose name will be returned.

If the passed sheet index does not exist, the method returns an empty name.

La indexación comienza en 0.

## Ejemplo

Obtener el nombre de la tercera hoja en el documento:

```
$sheetName:=VP Get sheet name("ViewProArea";2)
```

## Ver también

[VP Get sheet index](#)

## VP Get sheet options

VP Get sheet options ( *vpAreaName* : Text {; *sheet* : Integer } ) : Object

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre del área 4D View Pro en el formulario
sheet	Integer	->	Índice de la hoja (hoja actual si se omite)
Resultado	Objeto	<-	Objeto opciones de la hoja

## Descripción

The `VP Get sheet options` command returns an object containing the current sheet options of the `vpAreaName` area.

Pass the name of the 4D View Pro area in `vpAreaName`. If you pass a name that does not exist, an error is returned.

In the optional `sheet` parameter, you can designate a specific spreadsheet (counting begins at 0). Si se omite o si se pasa `vk current sheet`, se utiliza la hoja de cálculo actual.

## Objeto devuelto

The method returns an object containing the current values for all available sheet options. An option value may have been modified by the user or by the [VP SET SHEET OPTIONS](#) method.

To view the full list of the options, see [Sheet Options](#).

## Ejemplo

```
$options:=VP Get sheet options("ViewProArea")
If($options.colHeaderVisible) //column headers are visible
  ...
End if
```

## Ver también

[4D VIEW PRO SHEET OPTIONS](#)

[VP SET SHEET OPTIONS](#)

## VP Get show print lines

`VP Get show print lines ( vpAreaName : Text {; sheet : Integer } ) : Boolean`

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
sheet	Integer	<-	Índice de la hoja
Resultado	Booleano	<-	True si las líneas de impresión son visibles, de lo contrario False

## Descripción

The `VP Get show print lines` command returns `True` if the print preview lines are visible and `False` if they are hidden.

In `vpAreaName`, pass the name of the 4D View Pro area.

In `sheet`, pass the index of the target sheet. If `sheet` is omitted, the command applies to the current sheet.

La indexación comienza en 0.

## Ejemplo

The following code checks if preview lines are displayed or hidden in the document:

```

var $result : Boolean
$result:=VP Get show print lines("ViewProArea";1)

```

Ver también

[VP SET SHOW PRINT LINES](#)

## VP Get spans

VP Get spans ( *rangeObj* : Object ) : Object

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Objeto rango
Resultado	Objeto	<-	Objeto de celdas fusionadas en el rango definido

### Descripción

The `VP Get spans` command retrieves the cell spans in the designated *rangeObj*.

In *rangeObj*, pass a range of cell spans you want to retrieve. If *rangeObj* does not contain a cell span, an empty range is returned.

### Ejemplo

You want to center the text for the spanned cells in this document:

	First quarter			Second quarter		
	Jan	Feb	Mar	Apr	May	Jun
South area	John					
	Sahra					
	Dixie					

```

// Buscar todas las celdas fusionadas
$range:=VP Get spans(VP All("ViewProArea"))

//centrar el texto
$style:=New object("vAlign";vk vertical align center;"hAlign";vk horizontal align center)
VP SET CELL STYLE($range;$style)

```

Ver también

[VP ADD SPAN](#)

[VP REMOVE SPAN](#)

## VP Get stylesheet

VP Get stylesheet ( *vpAreaName* : Text ; *styleName* : Text { ; *scope* : Integer } ) : Object

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
styleName	Texto	->	Nombre del estilo
scope	Integer	->	Alcance objetivo (por defecto = hoja actual)
Resultado	Objeto	<-	Objeto hoja de estilo

## Descripción

The `VP Get stylesheet` command returns the `styleName` style sheet object containing the property values which have been defined.

In `vpAreaName`, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

In `styleName`, pass the name of the style sheet to get.

You can define where to get the style sheet in the optional `scope` parameter using the sheet index (counting begins at 0) or with the following constants:

- `vk current sheet`
- `vk workbook`

## Ejemplo

El código siguiente:

```
$style:=VP Get stylesheet("ViewProArea";"GreenDashDotStyle")
```

... will return the `GreenDashDotStyle` style object from the current sheet:

```
{
backColor:green,
borderBottom:{color:green,style:10},
borderLeft:{color:green,style:10},
borderRight:{color:green,style:10},
borderTop:{color:green,style:10}
}
```

## Ver también

[4D View Pro Style Objects and Style Sheets](#)

[VP ADD STYLESHEET](#)

[VP Get stylesheets](#)

[VP REMOVE STYLESHEET](#)

## VP Get stylesheets

`VP Get stylesheets ( vpAreaName : Text { ; scope : Integer } ) : Collection`

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
scope	Integer	->	Alcance objetivo (por defecto = hoja actual)
Resultado	Collection	<-	Colección de objetos de hojas de estilo

## Descripción

The `VP Get stylesheets` command returns the collection of defined style sheet objects from the designated *scope*.

In *vpAreaName*, pass the name property of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

You can define where to get the style sheets in the optional *scope* parameter using the sheet index (counting begins at 0) or with the following constants:

- `vk current sheet`
- `vk workbook`

## Ejemplo

The following code will return a collection of all the style objects in the current sheet:

```
$styles:=VP Get stylesheets("ViewProArea")
```

In this case, the current sheet uses two style objects:

```
[  
 {  
   backColor:green,  
   borderLeft:{color:green,style:10},  
   borderTop:{color:green,style:10},  
   borderRight:{color:green,style:10},  
   borderBottom:{color:green,style:10},  
   name:GreenDashDotStyle  
>,  
 {  
   backColor:red,  
   textIndent:10,  
   name:RedIndent  
>  
 ]
```

## Ver también

[VP ADD STYLESHEET](#)

[VP Get stylesheet](#)

[VP REMOVE STYLESHEET](#)

## VP Get value

`VP Get value ( rangeObj : Object ) : Object`

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto rango
Resultado	Objeto	<-	Objeto que contiene un valor de celda

### Descripción

The `VP Get value` command retrieves a cell value from a designated cell range.

In *rangeObj*, pass a range whose value you want to retrieve.

### Objeto devuelto

The object returned will contain the `value` property, and, in case of a js date value, a `time` property:

Propiedad	Tipo	Descripción
value	Integer, Real, Boolean, Text, Date	Value in the <i>rangeObj</i> (except- time)
time	Real	Time value (in seconds) if the value is of the js date type

If the object returned includes a date or time, it is treated as a datetime and completed as follows:

- time value - the date portion is completed as December 30, 1899 in dd/MM/yyyy format (30/12/1899)
- date value - the time portion is completed as midnight in HH:mm:ss format (00:00:00)

If *rangeObj* contains multiple cells or multiple ranges, the value of the first cell is returned. The command returns a null object if the cell is empty.

## Ejemplo

```
$cell:=VP Cell("ViewProArea";5;2)
$value:=VP Get value($cell)
If(Value type($value.value)=Is text)
    VP SET TEXT VALUE($cell;New object("value";Uppercase($value.value)))
End if
```

## Ver también

[VP Get values](#)  
[VP SET VALUE](#)  
[VP SET VALUES](#)

## VP Get values

VP Get values ( *rangeObj* : Object ) : Collection

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto rango
Resultado	Collection	<-	Colección de valores

## Descripción

The `VP Get values` command retrieves the values from the designated *rangeObj*.

In *rangeObj*, pass a range whose values you want to retrieve. If *rangeObj* includes multiple ranges, only the first range is used.

The collection returned by `VP Get values` contains a two-dimensional collection:

- Each element of the first-level collection represents a row and contains a subcollection of values
- Each subcollection contains cell values for the row. Los valores pueden ser Integer, Real, Boolean, Text, Null. If a value is a date or time, it is returned in an object with the following properties:

Propiedad	Tipo	Descripción
value	Fecha	Valor de la celda (excepto - time)
time	Real	Time value (in seconds) if the value is of the js date type

Dates or times are treated as a datetime and completed as follows:

- time value - the date portion is completed as December 30, 1899
- date value - the time portion is completed as midnight (00:00:00:000)

## Ejemplo

Quiere obtener los valores de C4 a G6:

	A	B	C	D	E	F	G
1							
2			1	2	3	FALSE	
3							
4			4	5		hello	world
5			6	7	8	9	
6			29/05/2019 0:00:42				
7							

```
$result:=VP Get values(VP Cells("ViewProArea";2;3;5;3))
// $result[0]=[4,5,null,hello,world]
// $result[1]=[6,7,8,9,null]
// $result[2]=[null,{time:42,value:2019-05-29T00:00:00.000Z},null,null,null]
```

Ver también

[VP Get formulas](#)

[VP Get value](#)

[VP SET FORMULAS](#)

[VP SET VALUES](#)

## VP Get workbook options

VP Get workbook options ( *vpAreaName* : Text ) : Object

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
Resultado	Objeto	<-	Objeto que contiene las opciones del libro de trabajo

Descripción

`VP Get workbook options` returns an object containing all the workbook options in *vpAreaName*

In *vpAreaName*, pass the name of the 4D View Pro area.

The returned object contains all the workbook options (default and modified ones), in the workbook.

The list of workbook options is referenced in [VP SET WORKBOOK OPTIONS](#)'s description.

Ejemplo

```
var $workbookOptions : Object
$workbookOptions:=VP Get workbook options("ViewProArea")
```

Ver también

[VP SET WORKBOOK OPTIONS](#)

I

## VP IMPORT DOCUMENT

VP IMPORT DOCUMENT ( *vpAreaName* : Text ; *filePath* : Text { ; *paramObj* : Object} )

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
<i>filePath</i>	Texto	->	Ruta de acceso del documento
<i>paramObj</i>	Objeto	->	Opciones de importación

## Descripción

The `VP IMPORT DOCUMENT` command imports and displays the document designated by *filePath* in the 4D View Pro area *vpAreaName*. The imported document replaces any data already inserted in the area.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

In *filePath*, pass the path and name of the document to be imported. Se soportan los siguientes formatos:

- Los documentos 4D View Pro (extensión ".4vp")
- Microsoft Excel (extensión ".xlsx")
- text documents (extension ".txt", ".csv", the document must be in utf-8)

If the document extension is not a recognized extension, such as `.4vp` or `.xlsx`, the document is considered a text document. You must pass a full path, unless the document is located at the same level as the Project folder, in which case you can just pass its name.

When importing a Microsoft Excel-formatted file into a 4D View Pro document, some settings may be lost. You can verify your settings with [this list from GrapeCity](#).

An error is returned if the *filePath* parameter is invalid, or if the file is missing or malformed.

The optional *paramObj* parameter allows you to define properties for the imported document:

Parámetros		Tipo	Descripción
<i>formula</i>		objeto	A callback method name to be launched when the import has completed. The method must use the <code>Formula</code> command. See <a href="#">Passing a callback method (formula)</a> .
<i>contraseña</i>		texto	Microsoft Excel only (optional) - The password used to protect a MS Excel document.
<i>csvOptions</i>		objeto	opciones de importación csv
	<i>range</i>	objeto	Cell range that contains the first cell where the data will be written. If the specified range is not a cell range, only the first cell of the range is used.
	<i>rowDelimiter</i>	texto	Delimitador de línea. If not present, the delimiter is automatically determined by 4D.
	<i>columnDelimiter</i>	texto	Delimitador de columna. Por defecto: ","

For more information on the CSV format and delimiter-separated values in general, see [this article on Wikipedia](#)

## Ejemplo 1

You want to import a default 4D View Pro document stored on the disk when the form is open:

```

C_TEXT($docPath)
If(Form event code=On VP Ready) //4D View Pro area loaded and ready
    $docPath:="C:\\Bases\\ViewProDocs\\MyExport.4VP"
    VP IMPORT DOCUMENT("VPArea";$docPath)
End if

```

## Ejemplo 2

You want to import a password protected Microsoft Excel document into a 4D View Pro area:

```

$o:=New object
$o.password:="excel123"

VP IMPORT DOCUMENT("ViewProArea";"c:\\tmp\\excefilefile.xlsx";$o)

```

## Ejemplo 3

You want to import a `.txt` file that uses a comma (",") as delimiter:

```

"Clark","Kent"
"Bruce","Wayne"
"Barry","Allen"
"Peter","Parker"
"Tony","Stark"

```

```

$params:=New object
$params.range:=VP Cells("ViewProArea";0;0;2;5)
VP IMPORT DOCUMENT("ViewProArea";"c:\\import\\my-file.txt";New object("csvOptions";$params))

```

Here's the result:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Clark	Kent											
2	Bruce	Wayne											
3	Barry	Allen											
4	Peter	Parker											
5	Tony	Stark											
6													
7													
8													
9													
10													
11													
12													
13													
14													

Ver también

[VP EXPORT DOCUMENT](#)  
[VP NEW DOCUMENT](#)

## VP IMPORT FROM OBJECT

VP IMPORT FROM OBJECT ( *vpAreaName* : Text { ; *viewPro* : Object} )

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
<i>viewPro</i>	Objeto	->	Objeto 4D View Pro

### Descripción

The `VP IMPORT FROM OBJECT` command imports and displays the *viewPro* 4D View Pro object in the *vpAreaName* 4D View Pro area. The imported object contents replaces any data already inserted in the area.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

In *viewPro*, pass a valid 4D View Pro object. This object can have been created using [VP Export to object](#) or manually. For more information on 4D View Pro objects, please refer to the [4D View Pro object](#) section.

An error is returned if the *viewPro* object is invalid.

### Ejemplo

You want to import a spreadsheet that was previously saved in an object field:

```
QUERY( [VPWorkBooks] ; [VPWorkBooks] ID=10 )
VP IMPORT FROM OBJECT("ViewProArea1"; [VPWorkBooks] SPBook)
```

### Ver también

[VP Export to object](#)

## VP INSERT COLUMNS

VP INSERT COLUMNS ( *rangeObj* : Object )

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Objeto rango

### Descripción

The `VP INSERT COLUMNS` command inserts columns into the *rangeObj*.

In *rangeObj*, pass an object containing a range of the starting column (the column which designates where the new column will be inserted) and the number of columns to insert. If the number of column to insert is omitted (not defined), a single column is inserted.

New columns are inserted on the left, directly before the starting column in the *rangeObj*.

### Ejemplo

Para insertar tres columnas antes de la segunda columna:

```
VP INSERT COLUMNS(VP Column("ViewProArea";1;3))
```

El resultado es:

## Before insertion

	A	B	C	D	E	F	G	H	I	J
1	The	quick	brown	fox	jumped	over	the	lazy	dog	
2										
3										

## After insertion

	A	B	C	D	E	F	G	H	I	J	K	L
1	The				quick	brown	fox	jumped	over	the	lazy	dog
2												
3												

Ver también

[VP DELETE COLUMNS](#)

[VP DELETE ROWS](#)

[VP INSERT ROWS](#)

## VP INSERT ROWS

VP INSERT ROWS ( *rangeObj* : Object )

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Objeto rango

### Descripción

The `VP INSERT ROWS` command inserts rows defined by the *rangeObj*.

In *rangeObj*, pass an object containing a range of the starting row (the row which designates where the new row will be inserted) and the number of rows to insert. If the number of rows to insert is omitted (not defined), a single row is inserted.

New rows are inserted directly before the first row in the *rangeObj*.

### Ejemplo

Para insertar 3 líneas antes de la primera línea:

```
VP INSERT ROWS(VP Row("ViewProArea";0;3))
```

El resultado es:

### Before insertion

	A	B	C	D	E	F	G	H	I	J
1	The	quick	brown	fox	jumped	over	the	lazy	dog	
2										
3										
4										
<										

### After insertion

	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4	The	quick	brown	fox	jumped	over	the	lazy	dog	
<										

Ver también

[VP DELETE COLUMNS](#)

[VP DELETE ROWS](#)

[VP INSERT COLUMNS](#)

## M

### VP MOVE CELLS

► Histórico

**VP MOVE CELLS ( *originRange* : Object ; *targetRange* : Object ; *options* : Object )**

Parámetros	Tipo		Descripción
<i>originRange</i>	Objeto	->	Rango de celdas desde donde copiar
<i>targetRange</i>	Objeto	->	Target range for the values, formatting and formulas
<i>options</i>	Objeto	->	Opciones adicionales

#### Descripción

The **VP MOVE CELLS** command moves or copies the values, style and formulas from *originRange* to *targetRange*.

*originRange* and *targetRange* can refer to different View Pro areas.

In *originRange*, pass a range object containing the values, style, and formula cells to copy or move. If *originRange* is a combined range, only the first one is used.

In *targetRange*, pass the range of cells where the cell values, style, and formulas will be copied or moved.

The *options* parameter has several properties:

Propiedad	Tipo	Descripción														
copy	Booleano	Determines if the values, formatting and formulas of the cells in <i>originRange</i> are removed after the command executes: <ul style="list-style-type: none"> <li>• <i>False</i> (default) to remove them</li> <li>• <i>True</i> to keep them</li> </ul>														
pasteOptions	Entero largo	Especifica lo que se pega. Valores posibles: <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Valor</th> <th>Descripción</th> </tr> </thead> <tbody> <tr> <td><code>vk clipboard options all</code> (por defecto)</td> <td>Pastes all data objects, including values, formatting, and formulas.</td> </tr> <tr> <td><code>vk clipboard options formatting</code></td> <td>Pega sólo el formato.</td> </tr> <tr> <td><code>vk clipboard options formulas</code></td> <td>Pegar sólo las fórmulas.</td> </tr> <tr> <td><code>vk clipboard options formulas and formatting</code></td> <td>Pega las fórmulas y el formato.</td> </tr> <tr> <td><code>vk clipboard options values</code></td> <td>Pega sólo los valores.</td> </tr> <tr> <td><code>vk clipboard options value and formatting</code></td> <td>Pastes the values and formatting.</td> </tr> </tbody> </table>	Valor	Descripción	<code>vk clipboard options all</code> (por defecto)	Pastes all data objects, including values, formatting, and formulas.	<code>vk clipboard options formatting</code>	Pega sólo el formato.	<code>vk clipboard options formulas</code>	Pegar sólo las fórmulas.	<code>vk clipboard options formulas and formatting</code>	Pega las fórmulas y el formato.	<code>vk clipboard options values</code>	Pega sólo los valores.	<code>vk clipboard options value and formatting</code>	Pastes the values and formatting.
Valor	Descripción															
<code>vk clipboard options all</code> (por defecto)	Pastes all data objects, including values, formatting, and formulas.															
<code>vk clipboard options formatting</code>	Pega sólo el formato.															
<code>vk clipboard options formulas</code>	Pegar sólo las fórmulas.															
<code>vk clipboard options formulas and formatting</code>	Pega las fórmulas y el formato.															
<code>vk clipboard options values</code>	Pega sólo los valores.															
<code>vk clipboard options value and formatting</code>	Pastes the values and formatting.															

The paste options defined in the [workbook options](#) are taken into account.

## Ejemplo

To copy the contents, values, formatting and formulas from an origin range:

```
var $originRange; $targetRange; $options : Object
$originRange:=VP Cells("ViewProArea"; 0; 0; 2; 5)
$targetRange:=VP Cells("ViewProArea"; 4; 0; 2; 5)
$options:=New object
$options.copy:=True
$options.pasteOptions:=vk clipboard options all
VP MOVE CELLS($originRange; $targetRange; $options)
```

## Ver también

[VP Copy to object](#)  
[VP PASTE FROM OBJECT](#)  
[VP SET WORKBOOK OPTIONS](#)

## N

## VP Name

VP Name ( *vpAreaName* : Text ; *rangeName* : Text { ; *scope* : Integer } ) : Object

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
rangeName	Texto	->	Nombre del rango existente
scope	Integer	->	Ubicación del rango (hoja actual si se omite)
Resultado	Objeto	<-	Rango de nombre

## Descripción

The `VP Name` command returns a new range object referencing a named range.

In `vpAreaName`, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

The `rangeName` parameter specifies an existing named cell range.

In the optional `scope` parameter, you can designate a specific spreadsheet where `rangeName` is defined. If omitted, the current spreadsheet is used by default. You can explicitly select the current spreadsheet or the entire workbook with the following constants:

- `vk current sheet`
- `vk workbook`

## Ejemplo

You want to give a value to the "Total" named range.

```
// name the B5 cell as Total
VP ADD RANGE NAME(VP Cell("ViewProArea";1;4);"Total")
$name:=VP Name("ViewProArea";" Total")
VP SET NUM VALUE($name;285;"$#,###.00")
```

## Ver también

[VP ADD RANGE NAME](#)  
[VP ALL](#)  
[VP Cell](#)  
[VP Cells](#)  
[VP Column](#)  
[VP Combine ranges](#)  
[VP Get names](#)  
[VP REMOVE NAME](#)  
[VP Row](#)

## VP NEW DOCUMENT

`VP NEW DOCUMENT ( vpAreaName : Text )`

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro

## Descripción

The `VP NEW DOCUMENT` command loads and displays a new, default document in the 4D View Pro form area object `vpAreaName`. The new empty document replaces any data already inserted in the area.

In `vpAreaName`, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

## Ejemplo

You want to display an empty document in the "myVPArea" form object:

```
VP NEW DOCUMENT("myVPArea")
```

Ver también

[VP IMPORT DOCUMENT](#)

---

## O

### VP Object to font

VP Object to font ( *fontObj* : Object ) : Text

Parámetros	Tipo		Descripción
font object	Objeto	->	Objeto fuente
Resultado	Texto	<-	Cadena de texto corto de la fuente

#### Descripción

The `VP Font to object` utility command returns an object from a font shorthand string.

In *fontObj*, pass an object containing the font properties. Se soportan las siguientes propiedades:

Propiedad	Tipo	Descripción	Valores posibles	Obligatorio
family	texto	Define la fuente.	todo tipo de familia de fuentes estándar o genérica. Ej. "Arial", "Helvetica", "serif", "arial,sans-serif"	Sí
size	texto	Define el tamaño de la fuente. The line-height can be added to the font-size: font-size/line-height: Ex: "15pt/20pt"	un número con una de las siguientes unidades: <ul style="list-style-type: none"> <li>• "em", "ex", "%", "px", "cm", "mm", "in", "pt", "pc", "ch", "rem", "vh", "vw", "vmin", "vmax"</li> <li>• or one of the following:</li> <li>• <code>vk font size large</code></li> <li>• <code>vk font size larger</code></li> <li>• <code>vk font size x large</code></li> <li>• <code>vk font size xx large</code></li> <li>• <code>vk font size small</code></li> <li>• <code>vk font size smaller</code></li> <li>• <code>vk font size x small</code></li> <li>• <code>vk font size xx small</code></li> </ul>	Sí
style	texto	Estilo de fuente.	<ul style="list-style-type: none"> <li>• <code>vk font style italic</code></li> <li>• <code>vk font style oblique</code></li> </ul>	No
variant	texto	Especifica el tipo de letra en minúsculas.	<ul style="list-style-type: none"> <li>• <code>vk font variant small caps</code></li> </ul>	No
weight	texto	Define el grosor de la fuente.	<ul style="list-style-type: none"> <li>• <code>vk font weight 100</code></li> <li>• <code>vk font weight 200</code></li> <li>• <code>vk font weight 300</code></li> <li>• <code>vk font weight 400</code></li> <li>• <code>vk font weight 500</code></li> <li>• <code>vk font weight 600</code></li> <li>• <code>vk font weight 700</code></li> <li>• <code>vk font weight 800</code></li> <li>• <code>vk font weight 900</code></li> <li>• <code>vk font weight bold</code></li> <li>• <code>vk font weight bolder</code></li> <li>• <code>vk font weight lighter</code></li> </ul>	No

This object can be created with the [VP Font to object](#) command.

The returned shorthand string can be assigned to the "font" property of a cell with the [VP SET CELL STYLE](#), for example.

## Ejemplo

```
$cellStyle:=VP Get cell style($range)

$font:=VP Font to object($cellStyle.font)
$font.style:=vk font style oblique
$font.variant:=vk font variant small caps
$font.weight:=vk font weight bolder

$cellStyle.font:=VP Object to font($font)
// $cellStyle.font contiene "bolder oblique small-caps 16pt arial"
```

Ver también

[4D View Pro Style Objects and Style Sheets](#)

[VP Font to object](#)

[VP SET CELL STYLE](#)

[VP SET DEFAULT STYLE](#)

## P

### VP PASTE FROM OBJECT

► Histórico

**VP PASTE FROM OBJECT** ( *rangeObj* : Object ; *dataObject* : Object {; *options* : Longint} )

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Objeto de rango de celda
<i>dataObject</i>	Objeto	->	Objeto que contiene los datos a pegar
<i>options</i>	Entero largo	->	Specifies what is pasted

#### Descripción

The **VP PASTE FROM OBJECT** command pastes the contents, style and formulas stored in *dataObject* to the *rangeObj* object.

In *rangeObj*, pass the cell range object where the values, formatting, and/or formula cells will be pasted. If *rangeObj* refers to more than one cell, only the first one is used.

In *dataObject*, pass the object that contains the cell data, formatting, and formulas to be pasted.

In the optional *options* parameter, you can specify what to paste in the cell range. Valores posibles:

Constante	Descripción
<code>vk clipboard options all</code>	Pastes all data objects, including values, formatting, and formulas.
<code>vk clipboard options formatting</code>	Pega sólo el formato.
<code>vk clipboard options formulas</code>	Pegar sólo las fórmulas.
<code>vk clipboard options formulas and formatting</code>	Pega fórmulas y formato.
<code>vk clipboard options values</code>	Pega sólo valores.
<code>vk clipboard options value and formatting</code>	Pega valores y formato.

The paste options defined in the [workbook options](#) are taken into account.

If *options* refers to a paste option not present in the copied object (e.g. formulas), the command does nothing.

#### Ejemplo

See example the example from [VP Copy to object](#)

#### Ver también

[VP Copy to object](#)

[VP MOVE CELLS](#)

[VP Get workbook options](#)

[VP SET WORKBOOK OPTIONS](#)

## VP PRINT

VP PRINT ( *vpAreaName* : Text { ; *sheet* : Integer } )

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
<i>sheet</i>	Integer	->	Índice de la hoja (hoja actual si se omite)

### Descripción

The `VP PRINT` command opens a print dialog window to print *vpAreaName*.

Pass the 4D View Pro area to be printed in *vpAreaName*. The command will open the system print dialog window where the printer can be specified and the page properties can be defined.

The properties defined in the print dialog window are for the printer paper, they are not the printing properties for the 4D View Pro area. Printing properties for 4D View Pro areas are defined using the [VP SET PRINT INFO](#) command. It is highly recommended that the properties for both the printer and the 4D View Pro area match, otherwise the printed document may not correspond to your expectations.

In the optional *sheet* parameter, you can designate a specific spreadsheet to print (counting begins at 0). Si se omite, la hoja actual se utiliza por defecto. You can explicitly select the current spreadsheet or entire workbook with the following constants:

- `vk current sheet`
- `vk workbook`

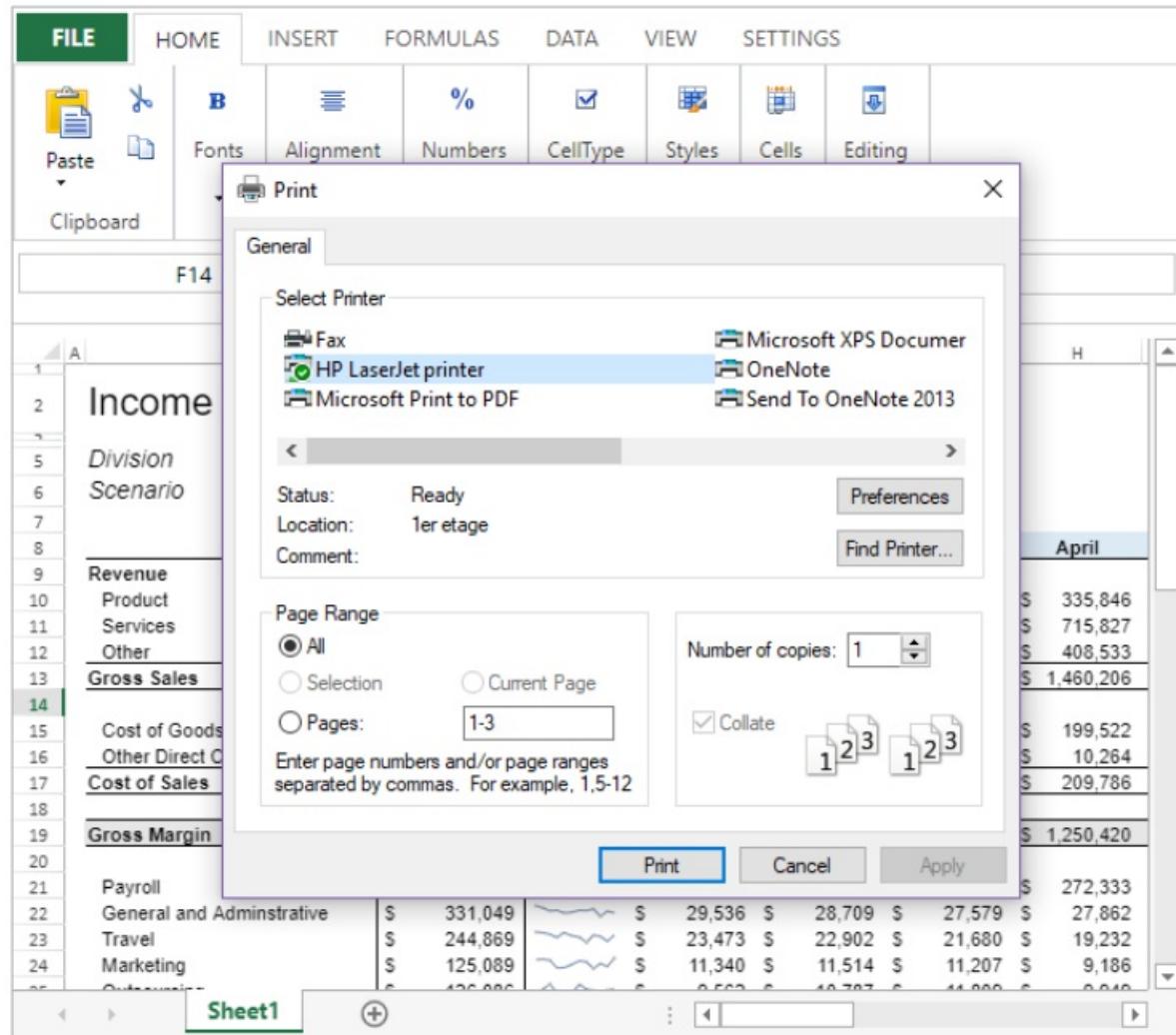
- 4D View Pro areas can only be printed with the `VP PRINT` command.
- Commands from the 4D Printing language theme are not supported by `VP PRINT`.
- This command is intended for individual printing by the final end user. For automated print jobs, it is advised to export the 4D View Pro area as a PDF with the [VP EXPORT DOCUMENT](#) method.

### Ejemplo

El código siguiente:

```
VP PRINT("myVPArea")
```

... abrirá una ventana de diálogo de impresión:



Ver también

[VP EXPORT DOCUMENT](#)  
[VP SET PRINT INFO](#)

R

## VP RECOMPUTE FORMULAS

**VP RECOMPUTE FORMULAS ( vpAreaName : Text )**

Parámetros	Tipo	Descripción
vpAreaName	Texto	-> Nombre de objeto formulario área 4D View Pro

### Descripción

The **VP RECOMPUTE FORMULAS** command immediately evaluates all formulas in *vpAreaName*. By default, 4D automatically computes formulas when they are inserted, imported, or exported. **VP RECOMPUTE FORMULAS** allows you to force the compute at any time (e.g. in case modifications are made to the formulas or if the formulas contain calls to the database). The command launches the execution of the **VP FLUSH COMMANDS** command to execute any stored commands and clear the command buffer, then calculates all formulas in the workbook.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

Be sure the **VP SUSPEND COMPUTING** command has not been executed before using **VP RECOMPUTE FORMULAS**, otherwise the command does nothing.

## Ejemplo

Para refrescar todas las fórmulas del libro de trabajo:

```
VP RECOMPUTE FORMULAS("ViewProArea")
```

## Ver también

[VP RESUME COMPUTING](#)

[VP SUSPEND COMPUTING](#)

## VP REMOVE NAME

VP REMOVE NAME ( *vpAreaName* : Text ; *name* : Text { ; *scope* : Integer } )

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
<i>name</i>	Texto	->	Name of the named range or named formula to remove
<i>scope</i>	Integer	->	Alcance objetivo (por defecto=hoja actual)

## Descripción

The `VP REMOVE NAME` command removes the named range or named formula passed in the *name* parameter in the defined *scope*.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

Pass the named range or named formula that you want to remove in *name*.

You can define where to remove the name in *scope* using either the sheet index (counting begins at 0) or the following constants:

- `vk current sheet`
- `vk workbook`

## Ejemplo

```
$range:=VP Cell("ViewProArea";0;0)
VP ADD RANGE NAME("Total1";$range)

VP REMOVE NAME("ViewProArea";"Total1")
$formula:=VP Get formula by name("ViewProArea";"Total1")
//$formula=null
```

## Ver también

[VP Name](#)

## VP REMOVE SHEET

VP REMOVE SHEET ( *vpAreaName* : Text ; *index*: Integer )

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
<i>index</i>	Integer	->	Índice de la hoja a eliminar

Ver también

[VP ADD SHEET](#)

## Descripción

The `VP REMOVE SHEET` command removes the sheet with the specified *index* from the document loaded in *vpAreaName*.

In *vpAreaName*, pass the name of the 4D View Pro area.

In *index*, pass the index of the sheet to remove. If the passed *index* does not exist, the command does nothing.

La indexación comienza en 0.

## Ejemplo

El documento tiene actualmente tres hojas:



Eliminar la tercera hoja:

```
VP REMOVE SHEET("ViewProArea";2)
```



## VP REMOVE SPAN

`VP REMOVE SPAN ( rangeObj : Object )`

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto rango

## Descripción

The `VP REMOVE SPAN` command removes the span from the cells in *rangeObj*.

In *rangeObj*, pass a range object of the cell span. The spanned cells in the range are divided into individual cells.

## Ejemplo

Para eliminar todas las fusiones de celdas de este documento:

	First quarter			Second quarter		
	Jan	Feb	Mar	Apr	May	Jun
South area	John					
	Sahra					
	Dixie					

```
//find all cell spans
$span:=VP Get spans(VP All("ViewProArea"))

//remove the cell spans
VP REMOVE SPAN($span)
```

Resultado:

		First quart		Second qu			
		Jan	Feb	Mar	Apr	May	Jun
South area	John						
	Sahra						
	Dixie						

Ver también

[VP ADD SPAN](#)  
[VP Get spans](#)

## VP REMOVE STYLESHEET

VP REMOVE STYLESHEET ( *vpAreaName* : Text ; *styleName* : Text { ; *scope* : Integer } )

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
<i>styleName</i>	Texto	->	Nombre del estilo a eliminar
<i>scope</i>	Integer	->	Alcance objetivo (por defecto = hoja actual)

### Descripción

The `VP REMOVE STYLESHEET` command removes the style sheet passed in the *styleName* from the *vpAreaName*.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

Pass the style sheet to remove in the *styleName* parameter.

You can define where to remove the style in the optional *scope* parameter using the sheet index (counting begins at 0) or with the following constants:

- `vk current sheet`
- `vk workbook`

### Ejemplo

To remove the *GreenDashDotStyle* style object from the current sheet:

```
VP REMOVE STYLESHEET("ViewProArea";"GreenDashDotStyle")
```

Ver también

[VP ADD STYLESHEET](#)  
[VP Get stylesheet](#)  
[VP Get stylesheets](#)

## VP RESET SELECTION

**VP RESET SELECTION ( *vpAreaName* : Text { ; *sheet* : Integer } )**

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
<i>sheet</i>	Integer	->	Índice de la hoja (hoja actual si se omite)

## Descripción

The **VP RESET SELECTION** command deselects all cells, resulting in no current selection or visible active cell.

A default active cell (cell A1) remains defined for 4D View Pro commands.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

In the optional *sheet* parameter, you can designate a specific spreadsheet where the range will be defined (counting begins at 0). If omitted, the current spreadsheet is used by default. You can explicitly select the current spreadsheet with the following constant:

- `vk current sheet`

## Ejemplo

You want to deselect all cells (the active cell and any selected cells):

```
VP RESET SELECTION("myVPArea")
```

## Ver también

[VP ADD SELECTION](#)

[VP Get active cell](#)

[VP Get selection](#)

[VP SET ACTIVE CELL](#)

[VP SET SELECTION](#)

[VP SHOW CELL](#)

## VP RESUME COMPUTING

**VP RESUME COMPUTING ( *vpAreaName* : Text )**

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro

## Descripción

The **VP RESUME COMPUTING** command restarts the calculation of formulas in *vpAreaName*.

The command reactivates the calculation service in 4D View Pro. Any formulas impacted by changes made while calculations were suspended are updated, and formulas added after **VP RESUME COMPUTING** is executed are calculated.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

The 4D View Pro calculation service maintains a counter of suspend/resume actions. Therefore, each execution of **VP RESUME COMPUTING** must be balanced by a corresponding execution of the **VP SUSPEND COMPUTING** command.

## Ejemplo

See example in [VP SUSPEND COMPUTING](#).

Ver también

[VP RECOMPUTE FORMULAS](#)

[VP SUSPEND COMPUTING](#)

## VP Row

VP Row ( *vpAreaName* : Text; *row* : Integer { ; *rowCount* : Integer { ; *sheet* : Integer } } ) : Object

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
<i>row</i>	Integer	->	Índice de la línea
<i>rowCount</i>	Integer	->	Número de líneas
<i>sheet</i>	Integer	->	Índice de la hoja (hoja actual si se omite)
Resultado	Objeto	<-	Rango de línea(s)

### Descripción

The `VP Row` command returns a new range object referencing a specific row or rows.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

The *column* parameter defines the first column of the column range. Pass the row index (counting begins at 0) in this parameter. Pass the column index (counting begins at 0) in this parameter.

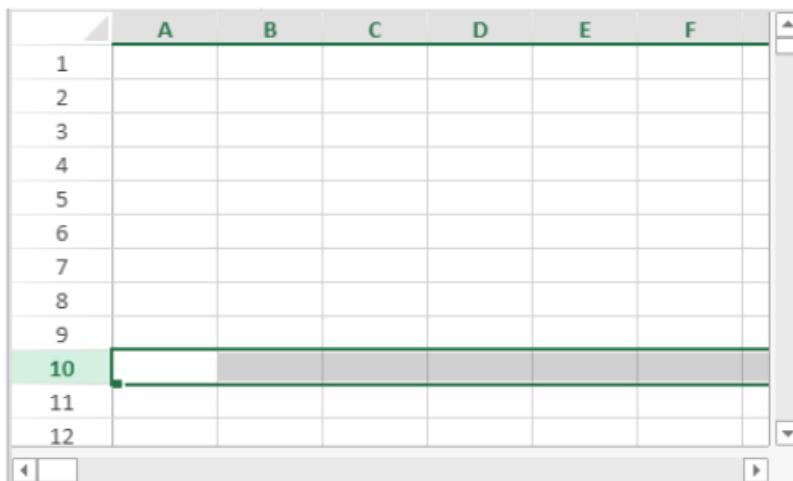
In the *row* parameter, you can define the row(s) of the cell range's position. *rowCount* must be greater than 0. Pass the row index (counting begins at 0) in this parameter.

In the optional *sheet* parameter, you can designate a specific spreadsheet where the range will be defined (counting begins at 0). If not specified, the current spreadsheet is used by default. You can explicitly select the current spreadsheet with the following constant:

- `vk current sheet`

### Ejemplo

You want to define a range object for the row shown below (on the current spreadsheet):



Puede escribir:

```
$row:=VP Row("ViewProArea";9) // línea 10
```

[Ver también](#)

VP All

VP Cell

## VP Cells

## VP Column

VP Combi

VB ROW AUTOEDIT

MR. DODGE MURKOFF / 600-0000-0000-0000

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto rango

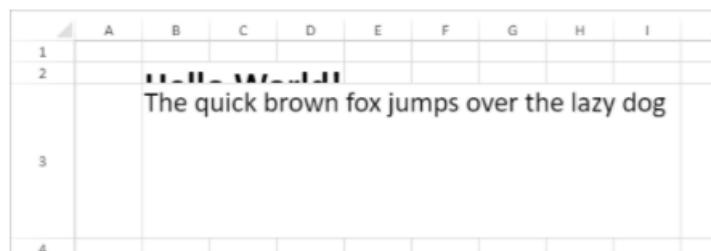
### Descripción

The **VP\_ROW\_AUTOFIT** command automatically sizes the row(s) in *rangeObj* according to their contents.

In `rangeObj`, pass a range object containing a range of the rows whose size will be automatically handled.

## Ejemplo

The following rows don't correctly display the text:



```
VP ROW AUTOFIT(VP Row("ViewProArea";1;2))
```

## Resultado:



[Ver también](#)

## VP Column autofit

## VP Run offscreen area

VP Run offscreen area ( parameters : Object ) : Mixed

Parámetros	Tipo		Descripción
parameters	Objeto	->	Objeto que contiene los atributos del área fuera de pantalla
Resultado	Mixed	<-	.result property of the .onEvent object, or Null if does not return a value

## Descripción

The `VP Run offscreen area` command creates an offscreen area in memory which can be used to process 4D View Pro area commands and functions.

In `parameters` object, pass any of the following optional properties. These properties will be available through the `This` command within the `onEvent` method and reference the instance:

Propiedad	Tipo	Descripción
area	texto	El nombre del área fuera de la pantalla. If omitted or null, a generic name is assigned (e.g., "OffscreenArea1").
onEvent	objet (fórmula)	<p>A callback method that will be launched when the offscreen area is ready. Puede ser:</p> <ul style="list-style-type: none"> <li>• una función <code>onEvent</code> de una clase, o</li> <li>• un objeto <code>Formula</code></li> </ul> <p>By default, the callback method is called on the <code>On VP Ready</code>, <code>On Load</code>, <code>On Unload</code>, <code>On End URL Loading</code>, <code>On URL Loading Error</code>, <code>On VP Range Changed</code>, or <code>On Timer</code> events.</p> <p>The callback method can be used to access the <a href="#">4D View Pro form object variable</a>.</p>
autoQuit	booleano	True (default value) if the command must stop the formula execution when the <code>On End URL Loading</code> or <code>On URL Loading Error</code> events occur. If false, you must use the <code>CANCEL</code> or <code>ACCEPT</code> commands in the <code>onEvent</code> callback method.
timeout	number	Maximum time (expressed in seconds) before the area automatically closes if no event is generated. Si se fija en 0, no se aplica ninguna limitación. Valor por defecto: 60
result	mixto	Resultado del procesamiento (si hay)
<customProperty>	mixto	Any custom attribute to be available in the <code>onEvent</code> callback method.

The following property is automatically added by the command if necessary:

Propiedad	Tipo	Descripción
timeoutReached	booleano	Added with true value if timeout has been exceeded

The offscreen area is only available during the execution of the `VP Run offscreen area` command. It will automatically be destroyed once execution has ended.

The following commands can be used in the callback method:

- `ACCEPT`
- `CANCEL`
- `SET TIMER`
- `WA Evaluate JavaScript`
- `WA EXECUTE JAVASCRIPT FUNCTION`

## Ejemplo 1

You want to create an offscreen 4D View Pro area and get the value of a cell:

```

// cs.OffscreenArea class declaration
Class constructor ($path : Text)
  This.filePath:=$path

// This function will be called on each event of the offscreen area
Function onEvent()
  Case of
    :(FORM Event.code=On VP Ready)
      VP IMPORT DOCUMENT(This.area;This.filePath)
      This.result:=VP Get value(VP Cell(This.area;6;22))

      ALERT("The G23 cell contains the value: "+String(This.result))
  End case

```

The *OffscreenArea* callback method:

```

$0:=cs.OffscreenArea.new()
$result:=VP Run offscreen area($0)

```

## Ejemplo 2

You want to load a large document offscreen, wait for all calculations to complete evaluating, and export it as a PDF:

```

//cs.OffscreenArea class declaration
Class constructor ($pdfPath : Text)
  This.pdfPath:=$pdfPath
  This.autoQuit:=False
  This.isWaiting:=False

Function onEvent()
  Case of
    :(FORM Event.code=On VP Ready)
    // Document import
      VP IMPORT DOCUMENT(This.area;$largeDocument4VP)
      This.isWaiting:=True

    // Start a timer to verify if all calculations are finished.
    // If during this period the "On VP Range Changed" is thrown, the timer will be restarted
    // The time must be defined according to the computer configuration.
      SET TIMER(60)

    :(FORM Event.code=On VP Range Changed)
    // End of calculation detected. Restarts the timer
      If(This.isWaiting)
        SET TIMER(60)
      End if

    :(FORM Event.code=On Timer)
    // To be sure to not restart the timer if you call others 4D View command after this point
      This.isWaiting:=False

    // Stop the timer
      SET TIMER(0)

    // Start the PDF export
      VP EXPORT DOCUMENT(This.area;This.pdfPath;New object("formula";Formula(ACCEPT)))

    :(FORM Event.code=On URL Loading Error)
      CANCEL
  End case

```

The `OffscreenArea` callback method:

```
$o:=cs.OffscreenArea.new()  
$result:=VP Run offscreen area($o)
```

Ver también

[Blog post: End of document loading](#)

## S

### VP SET ACTIVE CELL

VP SET ACTIVE CELL ( *rangeObj* : Object)

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Objeto rango

Descripción

The `VP SET ACTIVE CELL` command defines a specified cell as active.

In *rangeObj*, pass a range containing a single cell as an object (see [VP Cell](#)). If *rangeObj* is not a cell range or contains multiple ranges, the first cell of the first range is used.

Ejemplo

To set the cell in column D, row 5 as the active cell:

```
$activeCell:=VP Cell("myVPArea";3;4)  
VP SET ACTIVE CELL($activeCell)
```

A	B	C	D	E	F	G
1						
2	Month	Area	Mr. Smith	Ms. Johnson	Ms. Williams	Mr. Jones
3	January	South	1898	1857	1651	1448
4		East	4859	4857	2548	4876
5		North	2458	1524	6150	4987
6		West	5787	1580	3975	4878
7	Total		15002	9818	14324	16189
8	February	South	6668	4374	17495	9999
9		East	5955	1677	7944	9400
10		North	1000	6722	2195	2777
11		West	6896	8355	7195	2058
12	Total		20519	21128	34829	24234
13	March	South	2577	2000	6185	2704
14		East	4859	4857	2548	4876
15		North	2458	1524	6150	4987
16		West	5787	1580	3975	4878
17	Total		15681	9961	18858	17445

Ver también

[VP ADD SELECTION](#)

[VP Get active cell](#)

[VP Get selection](#)

[VP RESET SELECTION](#)

[VP SET SELECTION](#)

[VP SHOW CELL](#)

## VP SET ALLOWED METHODS

VP SET ALLOWED METHODS ( *methodObj* : Object)

Parámetros	Tipo		Descripción
methodObj	Objeto	->	Métodos permitidos en las áreas 4D View Pro

### Compatibilidad

For greater flexibility, it is recommended to use the [VP SET CUSTOM FUNCTIONS](#) command which allows you to designate 4D formulas that can be called from 4D View Pro areas. As soon as [VP SET CUSTOM FUNCTIONS](#) is called, [VP SET ALLOWED METHODS](#) calls are ignored. 4D View Pro also supports 4D's generic [SET ALLOWED METHODS](#) command if neither [VP SET CUSTOM FUNCTIONS](#) nor [VP SET ALLOWED METHODS](#) are called, however using the generic command is not recommended.

### Descripción

The [VP SET ALLOWED METHODS](#) command designates the project methods that can be called in 4D View Pro formulas. This command applies to all 4D View Pro areas initialized after its call during the session. It can be called multiple times in the same session to initialize different configurations.

By default for security reasons, if you do not execute the [VP SET ALLOWED METHODS](#) command, no method call is allowed in 4D View Pro areas -- except if 4D's generic [SET ALLOWED METHODS](#) command was used (see compatibility note). Using an unauthorized method in a formula prints a #NAME? error in the 4D View Pro area. error en el área 4D View Pro.

In the *methodObj* parameter, pass an object in which each property is the name of a function to define in the 4D View Pro areas:

Propiedad			Tipo	Descripción
<functionName>			Objeto	Definición de la función personalizada. The <functionName> property name defines the name of the custom function to display in 4D View Pro formulas (no spaces allowed)
	method		Texto	(mandatory) Name of the existing 4D project method to allow
	parameters		Colección de objetos	Collection of parameters (in the order they are defined in the method).
	[ ].name		Texto	Name of a parameter to display for the <functionName> . Note: Parameter names must not contain space characters.
	[ ].type		Número	<p>Tipo de parámetro. Tipos soportados:</p> <ul style="list-style-type: none"> <li>• Is Boolean</li> <li>• Is date</li> <li>• Is Integer</li> <li>• Is object</li> <li>• Is real</li> <li>• Is text</li> <li>• Is time</li> </ul> <p>If omitted, by default the value is automatically sent with its type, except date or time values which are sent as an object (see <a href="#">Parameters</a> section). If type is Is object , the object has the same structure as the object returned by <a href="#">VP Get value</a> .</p>
	summary		Texto	Descripción de la función a mostrar en 4D View Pro
	minParams		Número	Número mínimo de parámetros
	maxParams		Número	Número máximo de parámetros. Passing a number higher than the length of parameters allows declaring "optional" parameters with default type

## Ejemplo

You want to allow two methods in your 4D View Pro areas:

```

C_OBJECT($allowed)
$allowed:=New object //parameter for the command

$allowed.Hello:=New object //create a first simple function named "Hello"
$allowed.Hello.method:="My_Hello_Method" //sets the 4D method
$allowed.Hello.summary:="Hello prints hello world"

$allowed.Byebye:=New object //create a second function with parameters named "Byebye"
$allowed.Byebye.method:="My_ByeBye_Method"
$allowed.Byebye.parameters:=New collection
$allowed.Byebye.parameters.push(New object("name";"Message";"type";Is text))
$allowed.Byebye.parameters.push(New object("name";"Date";"type";Is date))
$allowed.Byebye.parameters.push(New object("name";"Time";"type";Is time))
$allowed.Byebye.summary:="Byebye prints a custom timestamp"
$allowed.Byebye.minParams:=3
$allowed.Byebye.maxParams:=3

VP SET ALLOWED METHODS($allowed)

```

After this code is executed, the defined functions can be used in 4D View Pro formulas:

	A	B	C	D	E
1	=BYEBYE(				
2		BYEBYE(Message; Date; Time)			
3			Summary		
4				Byebye prints a custom timestamp	
5					

In 4D View Pro formulas, function names are automatically displayed in uppercase.

Ver también

[4D functions](#)

[VP SET CUSTOM FUNCTIONS](#)

## VP SET BOOLEAN VALUE

VP SET BOOLEAN VALUE ( *rangeObj* : Object ; *boolValue* : Boolean )

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Objeto rango
<i>boolValue</i>	Booleano	->	Valor del booleano a definir

### Descripción

The `VP SET BOOLEAN VALUE` command assigns a specified boolean value to a designated cell range.

In *rangeObj*, pass a range of the cell(s) (created for example with `VP Cell` or `VP Column`) whose value you want to specify. If *rangeObj* includes multiple cells, the value specified will be repeated in each cell.

The *boolValue* parameter allows you to pass the boolean value ( True or False) that will be assigned to the *rangeObj*.

### Ejemplo

```

//Set the cell value as False
VP SET BOOLEAN VALUE(VP Cell("ViewProArea";3;2);False)

```

Ver también

[VP SET VALUE](#)

## VP SET BORDER

**VP SET BORDER** ( *rangeObj* : Object ; *borderStyleObj* : Object ; *borderPosObj* : Object )

Parámetros	Tipo	Descripción
<i>rangeObj</i>	Objeto	-> Objeto rango
<i>borderStyleObj</i>	Objeto	-> Objeto que contiene el estilo de línea de borde
<i>borderPosObj</i>	Objeto	-> Objeto que contiene la posición del borde

### Descripción

The **VP SET BORDER** command applies the border style(s) defined in *borderStyleObj* and *borderPosObj* to the range defined in the *rangeObj*.

In *rangeObj*, pass a range of cells where the border style will be applied. If the *rangeObj* contains multiple cells, borders applied with **VP SET BORDER** will be applied to the *rangeObj* as a whole (as opposed to the **VP SET CELL STYLE** command which applies borders to each cell of the *rangeObj*). If a style sheet has already been applied, **VP SET BORDER** will override the previously applied border settings for the *rangeObj*.

The *borderStyleObj* parameter allows you to define the style for the lines of the border. The *borderStyleObj* supports the following properties:

Propiedad	Tipo	Descripción	Valores posibles
<i>color</i>	texto	Define el color del borde. Por defecto = black.	CSS color "#rrggbb" syntax (preferred syntax), CSS color "rgb(r,g,b)" syntax (alternate syntax), CSS color name (alternate syntax)
<i>style</i>	Integer	Define el estilo del borde. Por defecto = empty.	<ul style="list-style-type: none"><li>• <code>vk line style dash dot</code></li><li>• <code>vk line style dash dot dot</code></li><li>• <code>vk line style dashed</code></li><li>• <code>vk line style dotted</code></li><li>• <code>vk line style double</code></li><li>• <code>vk line style empty</code></li><li>• <code>vk line style hair</code></li><li>• <code>vk line style medium</code></li><li>• <code>vk line style medium dash dot</code></li><li>• <code>vk line style medium dash dot dot</code></li><li>• <code>vk line style medium dashed</code></li><li>• <code>vk line style slanted dash dot</code></li><li>• <code>vk line style thick</code></li><li>• <code>vk line style thin</code></li></ul>

You can define the position of the *borderStyleObj* (i.e., where the line is applied) with the *borderPosObj*:

Propiedad	Tipo	Descripción
all	booleano	Estilo de la línea de borde aplicado a todos los bordes.
left	booleano	Estilo de la línea de borde aplicado al borde izquierdo.
top	booleano	Estilo de la línea de borde aplicado al borde superior.
right	booleano	Estilo de la línea de borde aplicado al borde derecho.
bottom	booleano	Estilo de la línea de borde aplicado al borde inferior.
outline	booleano	Estilo de línea de borde aplicado únicamente a los bordes exteriores.
inside	booleano	Estilo de la línea de borde aplicado únicamente a los bordes interiores.
innerHorizontal	booleano	Border line style applied to inner horizontal borders only.
innerVertical	booleano	Border line style applied to inner vertical borders only.

## Ejemplo 1

This code produces a border around the entire range:

```
$border:=New object("color";"red";"style";vk line style thick)
$option:=New object("outline";True)
VP SET BORDER(VP Cells("ViewProArea";1;1;3;3);$border;$option)
```

A screenshot of a spreadsheet application showing a 4x5 grid of cells. The columns are labeled A through E and the rows are labeled 1 through 6. A red border is drawn around the range B2:D4, which contains the values 1, 2, and 3 respectively. The rest of the grid is empty.

## Ejemplo 2

This code demonstrates the difference between `VP SET BORDER` and setting borders with the `VP SET CELL STYLE` command:

```
// Definir los bordes con VP SET BORDER
$border:=New object("color";"red";"style";vk line style thick)
$option:=New object("outline";True)
VP SET BORDER(VP Cells("ViewProArea";1;1;3;3);$border;$option)

// // Definir los bordes con VP SET CELL STYLE
$cellStyle:=New object
$cellStyle.borderBottom:=New object("color";"blue";"style";vk line style thick)
$cellStyle.borderRight:=New object("color";"blue";"style";vk line style thick)
VP SET CELL STYLE(VP Cells("ViewProArea";4;4;3;3);$cellStyle)
```

A screenshot of a spreadsheet application showing a 8x8 grid of cells. The columns are labeled A through H and the rows are labeled 1 through 8. A red border is drawn around the range B2:D4. On the bottom row (row 5), there are blue borders on the right edge from column E to H. On the rightmost column (column H), there are blue borders on the bottom edge from row 5 to 8. The rest of the grid is empty.

Ver también

## VP SET CELL STYLE

### VP SET CELL STYLE

VP SET CELL STYLE ( *rangeObj* : Object ; *styleObj* : Object)

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Objeto rango
<i>styleObj</i>	Objeto	->	Objeto style

#### Descripción

The `VP SET CELL STYLE` command applies the style(s) defined in the *styleObj* to the cells defined in the *rangeObj*.

In *rangeObj*, pass a range of cells where the style will be applied. If the *rangeObj* contains multiple cells, the style is applied to each cell.

Borders applied with `VP SET CELL STYLE` will be applied to each cell of the *rangeObj*, as opposed to the `VP SET BORDER` command which applies borders to the *rangeObj* as a whole.

The *styleObj* parameter lets you pass an object containing style settings. You can use an existing style sheet or create a new style. If the *styleObj* contains both an existing style sheet and additional style settings, the existing style sheet is applied first, followed by the additional settings.

To remove a style and revert to the default style settings (if any), pass a NULL value:

- giving the *styleObj* parameter a NULL value will remove any style settings from the *rangeObj*,
- giving an attribute a NULL value will remove this specific attribute from the *rangeObj*.

For more information about style objects and style sheets, see the [Style Objects](#) paragraph.

#### Ejemplo

```
$style:=New object
$style.font:="8pt Arial"
$style.backColor:="Azure"
$style.foreColor:="red"
$style.hAlign:=1
$style.isVerticalText:=True
$style.borderBottom:=New object("color";"#800080";"style";vk line style thick)
$style.backgroundImage:=Null //eliminar un atributo específico

VP SET CELL STYLE(VP Cell("ViewProArea";1;1);$style)
```

	A	B	C
1	Hello		
2	World		
3			

#### Ver también

[VP ADD STYLESHEET](#)

[VP Font to object](#)  
[VP Get cell style](#)  
[VP Object to font](#)  
[VP SET BORDER](#)  
[VP SET DEFAULT STYLE](#)

## VP SET COLUMN ATTRIBUTES

VP SET COLUMN ATTRIBUTES ( *rangeObj* : Object ; *propertyObj* : Object)

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Objeto rango
<i>propertyObj</i>	Objeto	->	Objeto que contiene las propiedades de columna

### Descripción

The `VP SET COLUMN ATTRIBUTES` command applies the attributes defined in the *propertyObj* to the columns in the *rangeObj*.

In *rangeObj*, pass an object containing a range. If the range contains both columns and rows, attributes are applied only to the columns.

The *propertyObj* parameter lets you specify the attributes to apply to the columns in the *rangeObj*. Estos atributos son:

Propiedad	Tipo	Descripción
ancho	number	Column width expressed in pixels
pageBreak	booleano	True to insert a page break before the first column of the range, else false
visible	booleano	True si la columna es visible, si no, false
redimensionable	booleano	True si la columna puede redimensionarse, si no, false
header	texto	Texto del encabezado de la columna

### Ejemplo

To change the size of the second column and set the header, you write:

```
C_OBJECT($column;$properties)  
  
$column:=VP_Column("ViewProArea";1) //column B  
$properties:=New object("width";100;"header";"Hello World")  
  
VP_SET_COLUMN_ATTRIBUTES($column;$properties)
```

	A	Hello World	C	D	E	F	G	H	I
1	The	quick	brown	fox	jumped	over	the	lazy	dog

### Ver también

[VP Column](#)  
[VP Get column attributes](#)  
[VP Get row attributes](#)  
[VP SET ROW ATTRIBUTES](#)

## VP SET COLUMN COUNT

VP SET COLUMN COUNT ( *vpAreaName* : Text , *columnCount* : Integer { , *sheet* : Integer } )

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
columnCount	Integer	->	Número de columnas
sheet	Integer	->	Índice de la hoja (hoja actual si se omite)

## Descripción

The `VP SET COLUMN COUNT` command defines the total number of columns in `vpAreaName`.

In `vpAreaName`, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

Pass the total number of columns in the `columnCount` parameter. `columnCount` must be greater than 0.

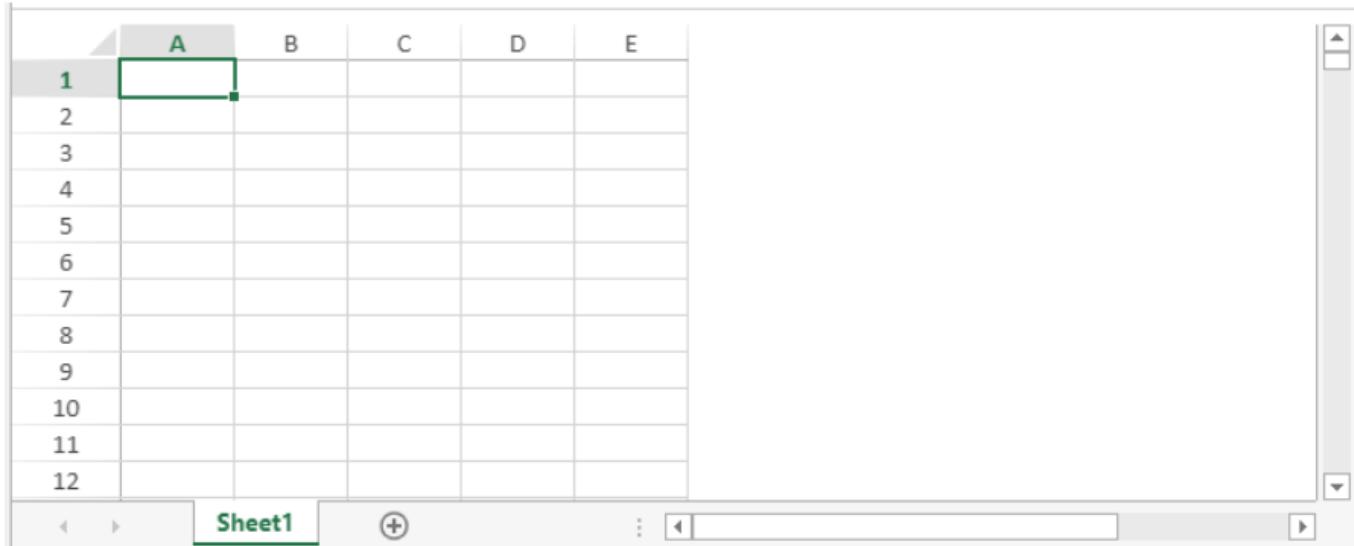
In the optional `sheet` parameter, you can designate a specific spreadsheet where the `columnCount` will be applied (counting begins at 0). If omitted, the current spreadsheet is used by default. You can explicitly select the current spreadsheet with the following constant:

- `vk current sheet`

## Ejemplo

The following code defines five columns in the 4D View Pro area:

```
VP SET COLUMN COUNT("ViewProArea";5)
```



## Ver también

[VP Get column count](#)

[VP Get row count](#)

[VP SET ROW COUNT](#)

## VP SET CURRENT SHEET

`VP SET CURRENT SHEET ( vpAreaName : Text ; index : Integer )`

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
index	Integer	<-	Índice de la nueva hoja actual

## Descripción

The `VP SET CURRENT SHEET` command sets the current sheet in `vpAreaName`. The current sheet is the selected sheet in the document.

In `vpAreaName`, pass the name of the 4D View Pro area.

In `index`, pass the index of the sheet to be set as current sheet. If the index passed is inferior to 0 or exceeds the number of sheets, the command does nothing.

La indexación comienza en 0.

## Ejemplo

La hoja actual del documento es la primera hoja:



Define la hoja actual en la tercera hoja:

```
VP SET CURRENT SHEET("ViewProArea";2)
```



Ver también

[VP Get current sheet](#)

## VP SET CUSTOM FUNCTIONS

`VP SET CUSTOM FUNCTIONS ( vpAreaName : Text ; formulaObj : Object )`

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
formulaObj	Objeto	->	Objeto formula

### Descripción

The `VP SET CUSTOM FUNCTIONS` command designates the 4D formulas that can be called directly from 4D View Pro formulas. Because custom functions are not stored in the document, `VP SET CUSTOM FUNCTIONS` must be executed in the `On Load` form event.

The formulas specified by `VP SET CUSTOM FUNCTIONS` appear in a pop-up menu when the first letter of their name is entered. See the [Formulas and Functions](#) page.

If `VP SET CUSTOM FUNCTIONS` is called multiple times for the same area, in the same session, only the last call is taken into account.

Pass the name of the 4D View Pro area in `vpAreaName`. If you pass a name that does not exist, an error is returned.

In the `formulaObj` parameter, pass an object containing the 4D formulas that can be called from 4D View Pro formulas as well as additional properties. Each `customFunction` property passed in `formulaObj` becomes the name of a function in the 4D View Pro area.

Propiedad			Tipo	Descripción
<customFunction>			Objeto	Definición de la función personalizada. <customFunction> defines the name of the custom function to display in 4D View Pro formulas (no spaces allowed)
	formula		Objeto	Objeto fórmula 4D (obligatorio). Ver el comando <a href="#">Formula</a> .
	parameters		Colección de objetos	Collection of parameters (in the order they are defined in the formula)
	[ ].name	Texto		Nombre del parámetro a mostrar en 4D View Pro
	[ ].type	Número		<p>Tipo de parámetro. Tipos soportados:</p> <ul style="list-style-type: none"> <li>• Is Boolean</li> <li>• Is date</li> <li>• Is Integer</li> <li>• Is object</li> <li>• Is real</li> <li>• Is text</li> <li>• Is time</li> </ul> <p>If type is omitted or if the default value (-1) is passed, the value is automatically sent with its type, except date or time values which are sent as an object (see <a href="#">Parameters</a> section).</p> <p>If type is Is object, the object has the same structure as the object returned by <a href="#">VP Get value</a>.</p>
	summary	Texto		Descripción de la Fórmula a mostrar en 4D View Pro
	minParams	Número		Número mínimo de parámetros
	maxParams	Número		Número máximo de parámetros. Passing a number higher than the length of <i>parameters</i> allows declaring "optional" parameters with default type

### ATENCIÓN

- As soon as `VP SET CUSTOM FUNCTIONS` is called, the methods allowed by the [VP SET ALLOWED METHODS](#) command (if any) are ignored in the 4D View Pro area.
- As soon as `VP SET CUSTOM FUNCTIONS` is called, the functions based upon `SET TABLE TITLES` and `SET FIELD TITLES` commands are ignored in the 4D View Pro area.

### Ejemplo

You want to use formula objects in a 4D View Pro area to add numbers, retrieve a customer's last name and gender:

```

Case of
:(FORM Event.code=On Load)

var $o : Object
$o:=New object

// Define "addnum" function from a method named "addnum"
$o.addnum:=New object
$o.addnum.formula:=Formula(addnum)
$o.addnum.parameters:=New collection
$o.addnum.parameters.push(New object("name";"num1";"type";Is Integer))
$o.addnum.parameters.push(New object("name";"num2";"type";Is Integer))

// Define "ClientLastName" function from a database field
$o.ClientLastName:=New object
$o.ClientLastName.formula:=Formula([Customers]lastname)
$o.ClientLastName.summary:="Lastname of the current client"

// Define "label" function from a 4D expression with one parameter
$o.label:=New object
$o.label.formula:=Formula(ds.Customers.get($1).label)
$o.label.parameters:=New collection
$o.label.parameters.push(New object("name";"ID";"type";Is Integer))

// Define "Title" function from a variable named "Title"
$o.Title:=New object
$o.Title.formula:=Formula(Title)

VP SET CUSTOM FUNCTIONS("ViewProArea";$o)

```

**End case**

Ver también

[VP SET ALLOWED METHODS](#)

## VP SET DATE TIME VALUE

**VP SET DATE TIME VALUE ( *rangeObj* : Object ; *dateValue* : Date ; *timeValue* : Time {; *formatPattern* : Text } )**

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Objeto rango
<i>dateValue</i>	Fecha	->	Valor date a definir
<i>timeValue</i>	Hora	->	Valor hora a definir
<i>formatPattern</i>	Texto	->	Formato del valor

### Descripción

The **VP SET DATE TIME VALUE** command assigns a specified date and time value to a designated cell range.

In *rangeObj*, pass a range of the cell(s) (created for example with [VP Cell](#) or [VP Column](#)) whose value you want to specify. If *rangeObj* includes multiple cells, the value specified will be repeated in each cell.

The *dateValue* parameter specifies a date value to be assigned to the *rangeObj*.

The *timeValue* parameter specifies a time value (expressed in seconds) to be assigned to the *rangeObj*.

The optional *formatPattern* defines a pattern for the *dateValue* and *timeValue* parameters. For information on patterns and formatting characters, please refer to the [Date and time formats](#) section.

## Ejemplo

```
//Definir el valor de la celda como la fecha y hora actuales  
VP SET DATE TIME VALUE(VP Cell("ViewProArea";6;2);Current time;Current date;vk pattern full date time)  
  
//Definir el valor de la celda como 18 de diciembre  
VP SET DATE TIME VALUE(VP Cell("ViewProArea";3;9);!2024-12-18!;?14:30:10?;vk pattern sortable date time)
```

Ver también

[4D View Pro cell format](#)

[VP SET DATE VALUE](#)

[VP SET TIME VALUE](#)

[VP SET VALUE](#)

## VP SET DATE VALUE

`VP SET DATE VALUE ( rangeObj : Object ; dateValue : Date { ; formatPattern : Text } )`

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto rango
dateValue	Fecha	->	Valor date a definir
formatPattern	Texto	->	Formato del valor

### Descripción

The `VP SET DATE VALUE` command assigns a specified date value to a designated cell range.

In `rangeObj`, pass a range of the cell(s) whose value you want to specify. If `rangeObj` includes multiple cells, the value specified will be repeated in each cell.

The `dateValue` parameter specifies a date value to be assigned to the `rangeObj`.

The optional `formatPattern` defines a pattern for the `dateValue` parameter. Pass any custom format or you can use one of the following constants:

Constante	Descripción	Configuración por defecto de US
<code>vk pattern long date</code>	Formato ISO 8601 para la fecha completa	"dddd, dd MMMM yyyy"
<code>vk pattern month day</code>	Formato ISO 8601 para el mes y el día	"MMMM dd"
<code>vk pattern short date</code>	Formato ISO 8601 corto para la fecha	"MM/dd/yyyy"
<code>vk pattern year month</code>	Formato ISO 8601 para el mes y el año	"yyyy MMMM"

For information on patterns and formatting characters, please refer to the [Date and time formats](#) section.

## Ejemplo

```
//Definir el valor de la celda para la fecha actual  
VP SET DATE VALUE(VP Cell("ViewProArea";4;2);Current date))  
  
//Definir el valor de la celda para una fecha específica con un formato designado  
VP SET DATE VALUE(VP Cell("ViewProArea";4;4);Date("12/25/94");"d/m/yy ")  
VP SET DATE VALUE(VP Cell("ViewProArea";4;6);!2005-01-15!;vk pattern month day)
```

Ver también

[4D View Pro cell format](#)

[VP SET DATE TIME VALUE](#)

[VP SET VALUE](#)

## VP SET DEFAULT STYLE

`VP SET DEFAULT STYLE ( vpAreaName : Text ; styleObj : Object { ; sheet : Integer } )`

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
styleObj	Objeto	->	Objeto style
sheet	Integer	->	Índice de la hoja (por defecto = hoja actual)

### Descripción

The `VP SET DEFAULT STYLE` command defines the style in the `styleObj` as the default style for a `sheet`.

In `vpAreaName`, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

The `styleObj` lets you pass an object containing style settings. You can use an existing style sheet or you can create a new style. For more information, see the [Style objects](#) paragraph.

In the optional `sheet` parameter, you can designate a specific spreadsheet where the style will be defined. If omitted, the current spreadsheet is used by default. You can explicitly select the current spreadsheet with the following constant:

- `vk current sheet`

### Ejemplo

```
$style:=New object
	style.hAlign:=vk horizontal align left
		style.font:="12pt papyrus"
		style.backColor:="#E6E6FA" //color mordado claro

VP SET DEFAULT STYLE("myDoc";$style)
```

A screenshot of a spreadsheet application showing a single row with data. The row has 8 columns, labeled A through H. Column A contains row numbers 1 through 8. Column B contains the text "Hello World!". The second row (row 2) is highlighted with a green background, and the cell B2 contains the text "Hello World!". The other cells in the row are empty.

1							
2							
3							
4							
5							
6							
7							
8							

### Ver también

[VP ADD STYLESHEET](#)

[VP Font to object](#)

[VP Get default style](#)

[VP Object to font](#)

[VP SET BORDER](#)

[VP SET CELL STYLE](#)

## VP SET FIELD

VP SET FIELD ( *rangeObj* : Object ; *field* : Pointer { ; *formatPattern* : Text } )

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Objeto rango
<i>campo</i>	Puntero	->	Referencia al campo en la estructura virtual
<i>formatPattern</i>	Texto	->	Formato del campo

## Descripción

The `VP SET FIELD` command assigns a 4D database virtual field to a designated cell range.

In *rangeObj*, pass a range of the cell(s) whose value you want to specify. In *rangeObj*, pass a range of the cell(s) (created for example with `VP Cell` or `VP Column`) whose value you want to specify.

The *field* parameter specifies a 4D database `virtual field` to be assigned to the *rangeObj*. The virtual structure name for *field* can be viewed in the formula bar. If any of the cells in *rangeObj* have existing content, it will be replaced by *field*.

The optional *formatPattern* defines a pattern for the *field* parameter. You can pass any valid `custom format`.

## Ejemplo

```
VP SET FIELD(VP Cell("ViewProArea";5;2);->[TableName]Field)
```

## Ver también

[VP SET VALUE](#)

## VP SET FORMULA

VP SET FORMULA ( *rangeObj* : Object ; *formula* : Text { ; *formatPattern* : Text } )

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Objeto rango
<i>formula</i>	Texto	->	Fórmula o método 4D
<i>formatPattern</i>	Texto	->	Formato del campo

## Descripción

The `VP SET FORMULA` command assigns a specified formula or 4D method to a designated cell range.

In *rangeObj*, pass a range of the cell(s) (created for example with `VP Cell` or `VP Column`) whose value you want to specify. In *rangeObj*, pass a range of the cell(s) (created for example with `VP Cell` or `VP Column`) whose value you want to specify.

The *formula* parameter specifies a formula or 4D method name to be assigned to the *rangeObj*. If a 4D method is used, it must be allowed with the `VP SET ALLOWED METHODS` command.

The optional *formatPattern* defines a `pattern` for the *formula*.

You remove the formula in *rangeObj* by replacing it with an empty string ("").

## Ejemplo 1

```
VP SET FORMULA(VP Cell("ViewProArea";5;2);"SUM($A$1:$C$10)")
```

## Ejemplo 2

Para eliminar la fórmula:

```
VP SET FORMULA(VP Cell("ViewProArea";5;2); "")
```

Ver también

[Cell format](#)

[VP Get Formula](#)

[VP SET FORMULAS](#)

[VP SET VALUE](#)

## VP SET FORMULAS

`VP SET FORMULAS ( rangeObj : Object ; formulasCol : Collection )`

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto de rango de celda
formulasCol	Collection	->	Colección de fórmulas

### Descripción

The `VP SET FORMULAS` command assigns a collection of formulas starting at the specified cell range.

In `rangeObj`, pass a range of the cell (created with [VP Cell](#)) whose formula you want to specify. If `rangeObj` includes multiple ranges, only the first range is used.

The `formulasCol` is a two-dimensional collection:

- The first-level collection contains subcollections of formulas. Cada subcolección define una línea.
- Each subcollection defines cell values for the row. Values must be text elements containing the formulas to assign to the cells. >If a 4D method is used, it must be allowed with the [VP SET ALLOWED METHODS](#) command.

You remove the formulas in `rangeObj` by replacing them with an empty string ("").

### Ejemplo 1

```
$formulas:=New collection
$formulas.push(New collection("MAX(B11,C11,D11)";"myMethod(G4)")) // First row
$formulas.push(New collection("SUM(B11:D11)";"AVERAGE(B11:D11)")) // Second row

VP SET FORMULAS(VP Cell("ViewProArea";6;3);$formulas) // Set the cells with the formulas
```

*myMethod:*

```
$0:=$1*3.33
```

The screenshot shows a spreadsheet interface with a toolbar at the top. The formula bar displays the formula `=Mymethod(G4)`. The main grid contains two sections: 'Production Cost Analysis' and 'Retail Price Calculations'. The 'Production Cost Analysis' section has columns for Product 1, Product 2, and Product 3, with data rows from 4 to 11. The 'Retail Price Calculations' section includes summary rows for Highest Prod Cost (\$4.42), Total Production Cost (\$11.38), and Average Product Cost (\$3.79). A 'Retail Price' value of \$14.72 is also shown.

## Ejemplo 2

Para eliminar las fórmulas:

```
$formulas:=New collection
$formulas.push(New collection("", ""))
$formulas.push(New collection("", ""))
VP SET FORMULAS(VP Cell("ViewProArea";0;0);$formulas) // Assign to cells
```

Ver también

[VP Get Formulas](#)  
[VP GET VALUES](#)  
[VP SET FORMULA](#)  
[VP SET VALUES](#)

## VP SET FROZEN PANES

`VP SET FROZEN PANES ( vpAreaName : Text ; paneObj : Object { ; sheet : Integer } )`

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
paneObj	Objeto	->	Object containing frozen column and row information
sheet	Integer	->	Índice de la hoja (hoja actual si se omite)

### Descripción

The `VP SET FROZEN PANES` commandsets the frozen status of the columns and rows in the `paneObj` so they are always displayed in the `vpAreaName`. Frozen columns and rows are fixed in place and do not move when the rest of the document is scrolled. A solid line is displayed to indicate that columns and rows are frozen. The location of the line depends on where the frozen column or row is on the sheet:

- Columns on the left or right: For columns on the left of the sheet, the line is displayed on the right side of the last frozen column. For columns on the right side of the sheet, the line is displayed on the left side of the first frozen column.
- Rows on the top or bottom: For rows at the top of the sheet, the line is displayed below the last frozen row. For rows at the bottom of the sheet, the line is displayed above the first frozen row.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

You can pass an object defining the columns and rows to freeze in the *paneObj* parameter. Setting the value of any of the column or row properties equal to zero resets (unfreezes) the property. If a property is set to less than zero, the command does nothing. Puede pasar:

Propiedad	Tipo	Descripción
columnCount	Integer	The number of frozen columns on the left of the sheet
trailingColumnCount	Integer	The number of frozen columns on the right of the sheet
rowCount	Integer	El número de líneas congeladas en la parte superior de la hoja
trailingRowCount	Integer	The number of frozen rows on the bottom of the sheet

In the optional *sheet* parameter, you can designate a specific spreadsheet where the range will be defined (counting begins at 0). If omitted, the current spreadsheet is used by default. You can explicitly select the current spreadsheet with the following constant:

- `vk current sheet`

## Ejemplo

You want to freeze the first three columns on the left, two columns on the right, and the first row:

```
C_OBJECT($panes)  
  
$panes:=New object  
$panes.columnCount:=3  
$panes.trailingColumnCount:=2  
$panes.rowCount:=1  
  
VP SET FROZEN PANES("ViewProArea";$panes)
```



## Ver también

[VP Get frozen panes](#)

## VP SET NUM VALUE

`VP SET NUM VALUE ( rangeObj : Object ; minValue : Number { ; formatPattern : Text } )`

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto rango
minValue	Número	->	Valor del número a definir
formatPattern	Texto	->	Formato del valor

## Descripción

The `VP SET NUM VALUE` command assigns a specified numeric value to a designated cell range.

In `rangeObj`, pass a range of the cell(s) (created for example with `VP Cell` or `VP Column`) whose value you want to specify. If `rangeObj` includes multiple cells, the value specified will be repeated in each cell.

The `newValue` parameter specifies a numeric value to be assigned to the `rangeObj`.

The optional `formatPattern` defines a `pattern` for the `newValue` parameter.

## Ejemplo

```
//Set the cell value to 2
VP SET NUM VALUE(VP Cell("ViewProArea";3;2);2)

//Set the cell value and format it in dollars
VP SET NUM VALUE(VP Cell("ViewProArea";3;2);12.356;"_($* #,##0.00_)")
```

## Ver también

[Cell format](#)  
[VP SET VALUE](#)

## VP SET PRINT INFO

`VP SET PRINT INFO ( vpAreaName : Text ; printInfo : Object { ; sheet : Integer } )`

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre del área 4D View Pro
printInfo	Objeto	->	Objeto que contiene los atributos de impresión
sheet	Integer	->	Índice de la hoja (hoja actual si se omite)

## Descripción

The `VP SET PRINT INFO` command defines the attributes to use when printing the `vpAreaName`.

Pass the name of the 4D View Pro area to print in `vpAreaName`. If you pass a name that does not exist, an error is returned.

You can pass an object containing definitions for various printing attributes in the `printInfo` parameter. To view the full list of the available attributes, see [Print Attributes](#).

In the optional `sheet` parameter, you can designate a specific spreadsheet to print (counting begins at 0). If omitted, the current spreadsheet is used by default. You can explicitly select the current spreadsheet with the following constant:

- `vk current sheet`

## Ejemplo

The following code will print a 4D View Pro area to a PDF document:

```

var $printInfo : Object

//declare print attributes object
$printInfo:=New object

//define print attributes
$printInfo.headerCenter:="&BS.H.I.E.L.D. &A Sales Per Region"
$printInfo.firstPageNumber:=1
$printInfo.footerRight:="page &P of &N"
$printInfo.orientation:=vk print page orientation landscape
$printInfo.centering:=vk print centering horizontal
$printInfo.columnStart:=0
$printInfo.columnEnd:=8
$printInfo.rowStart:=0
$printInfo.rowEnd:=24

$printInfo.showGridLine:=True

//Añadir logo corporativo
$printInfo.headerLeftImage:=logo.png
$printInfo.headerLeft:="&G"

$printInfo.showRowHeader:=vk print visibility hide
$printInfo.showColumnHeader:=vk print visibility hide
$printInfo.fitPagesWide:=1
$printInfo.fitPagesTall:=1

//imprimir documento PDF
VP SET PRINT INFO ("ViewProArea";$printInfo)

//exportar el PDF
VP EXPORT DOCUMENT("ViewProArea";"Sales2018.pdf";New object("formula";Formula(ALERT("PDF ready!"))))

```

El PDF:



## S.H.I.E.L.D. 2018 Sales Per Region

OrderDate	Region	Rep	Item	Units	UnitCost	Total
06-Jan-18	East	Stark	Pencil	95	\$1.99	\$189.05
23-Jan-18	Central	Rogers	Binder	50	\$19.99	\$999.50
09-Feb-18	Central	Banner	Pencil	36	\$4.99	\$179.64
26-Feb-18	Central	Romanoff	Pen	27	\$19.99	\$539.73
15-Mar-18	West	Barton	Pencil	56	\$2.99	\$167.44
01-Apr-18	East	Coulson	Binder	60	\$4.99	\$299.40
18-Apr-18	Central	Fury	Pencil	75	\$1.99	\$149.25
05-May-18	Central	Potts	Pencil	90	\$4.99	\$449.10
22-May-18	West	Jarvis	Pencil	32	\$1.99	\$63.68
08-Jun-18	East	Loki	Binder	60	\$8.99	\$539.40
25-Jun-18	Central	Odin	Pencil	90	\$4.99	\$449.10
06-Jul-18	East	Stark	Pencil	95	\$1.99	\$189.05
23-Jul-18	Central	Rogers	Binder	50	\$19.99	\$999.50
09-Aug-18	Central	Banner	Pencil	36	\$4.99	\$179.64
26-Aug-18	Central	Romanoff	Pen	27	\$19.99	\$539.73
15-Sep-18	West	Barton	Pencil	56	\$2.99	\$167.44
01-Oct-18	East	Coulson	Binder	60	\$4.99	\$299.40
18-Oct-18	Central	Fury	Pencil	75	\$1.99	\$149.25
05-Nov-18	Central	Potts	Pencil	90	\$4.99	\$449.10
22-Nov-18	West	Jarvis	Pencil	32	\$1.99	\$63.68
08-Dec-18	East	Loki	Binder	60	\$8.99	\$539.40
15-Dec-18	Central	Odin	Pencil	90	\$4.99	\$449.10
				1,342	\$155.78	\$8,050.58

page 1 of 1

## Ver también

[4D View Pro print attributes](#)[VP Convert to picture](#)[VP Get print info](#)[VP PRINT](#)

## VP SET ROW ATTRIBUTES

VP SET ROW ATTRIBUTES ( *rangeObj* : Object ; *propertyObj* : Object )

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Rango de líneas
<i>propertyObj</i>	Objeto	->	Object containing row properties

## Descripción

The `VP SET ROW ATTRIBUTES` command applies the attributes defined in the *propertyObj* to the rows in the *rangeObj*.

In the *rangeObj*, pass an object containing a range. If the range contains both columns and rows, attributes are applied only to the rows.

The *propertyObj* parameter lets you specify the attributes to apply to the rows in the *rangeObj*. Estos atributos son:

Propiedad	Tipo	Descripción
height	number	Altura de la línea expresada en píxeles
pageBreak	booleano	True to insert a page break before the first row of the range, else false
visible	booleano	True si la fila es visible, si no, false
redimensionable	booleano	True si la línea puede redimensionarse, si no, false
header	texto	Texto del encabezado de la línea

## Ejemplo

You want to change the size of the second row and set the header:

```
var $row; $properties : Object

$row:=VP Row("ViewProArea";1)
$properties:=New object("height";75;"header";"June")

VP SET ROW ATTRIBUTES($row;$properties)
```

	A	B	C	D	E	F	G	H	I
1	The	quick	brown	fox	jumped	over	the	lazy	dog
June									

## Ver también

[VP Get row attributes](#)  
[VP get column attributes](#)  
[VP SET ROW ATTRIBUTES](#)

## VP SET ROW COUNT

VP SET ROW COUNT ( *vpAreaName* : Text ; *rowCount* : Integer { ; *sheet* : Integer } )

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
<i>rowCount</i>	Integer	->	Número de líneas
<i>sheet</i>	Integer	->	Índice de la hoja (hoja actual si se omite)

## Descripción

The `VP SET ROW COUNT` command defines the total number of rows in *vpAreaName*.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

Pass the total number of rows in the *rowCount* parameter. *rowCount* must be greater than 0.

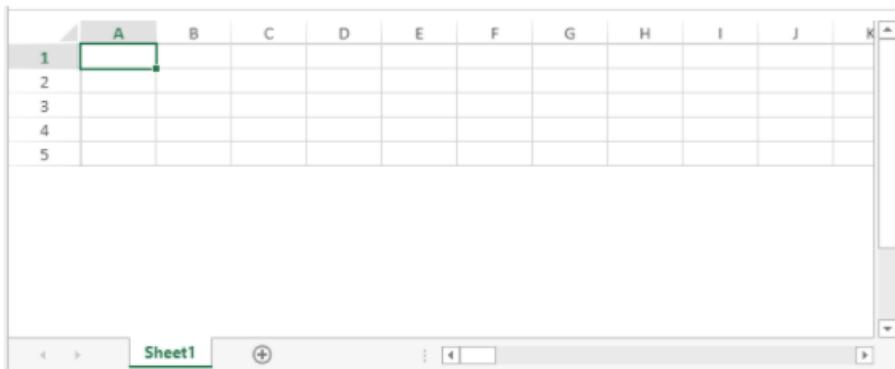
In the optional *sheet* parameter, you can designate a specific spreadsheet where the *rowCount* will be applied (counting begins at 0). If omitted, the current spreadsheet is used by default. You can explicitly select the current spreadsheet with the following constant:

- `vk current sheet`

## Ejemplo

The following code defines five rows in the 4D View Pro area:

```
VP SET ROW COUNT("ViewProArea";5)
```



Ver también

[VP Get column count](#)

[VP get row-count](#)

[VP SET COLUMN COUNT](#)

## VP SET SELECTION

**VP SET SELECTION ( *rangeObj* : Object )**

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Objeto rango de celdas

### Descripción

The **VP SET SELECTION** command defines the specified cells as the selection and the first cell as the active cell.

In *rangeObj*, pass a range object of cells to designate as the current selection.

### Ejemplo

```
$currentSelection:=VP Combine ranges(VP Cells("myVPArea";3;2;1;6);VP Cells("myVPArea";5;7;1;7))  
VP SET SELECTION($currentSelection)
```

	A	B	C	D	E	F	G
1							
2							
3	Month	Area	Mr. Smith	Ms. Johnson	Ms. Williams	Mr. Jones	M
4	January	South	1898	1857	1651	1448	
5		East	4859	4857	2548	4876	
6		North	2458	1524	6150	4987	
7		West	5787	1580	3975	4878	
8		Total	15002	9818	14324	16189	
9	February	South	6668	4374	17495	9999	
10		East	5955	1677	7944	9400	
11		North	1000	6722	2195	2777	
12		West	6896	8355	7195	2058	
13		Total	20519	21128	34829	24234	
14	March	South	2577	2000	6185	2704	
15		East	4859	4857	2548	4876	
16		North	2458	1524	6150	4987	
17		West	5787	1580	3975	4878	
		Total	15621	9961	18852	17115	

Ver también

[VP Get active cell](#)

[VP Get selection](#)

[VP RESET SELECTION](#)

[VP SET ACTIVE CELL](#)

[VP ADD SELECTION](#)

[VP SHOW CELL](#)

## VP SET SHEET COUNT

**VP SET SHEET COUNT ( *vpAreaName* : Text ; *number* : Integer )**

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
<i>number</i>	Integer	->	Número de hojas

### Descripción

The `VP SET SHEET COUNT` command sets the number of sheets in *vpAreaName*.

In *number*, pass a number corresponding to how many sheets the document will contain after the command is executed.

**Warning:** The command will delete sheets if the previous amount of sheets in your document is superior to the number passed. For example, if there are 5 sheets in your document and you set the sheet count to 3, the command will delete sheets number 4 and 5.

### Ejemplo

El documento tiene actualmente una hoja:



Para definir el número de hojas en 3:

```
VP SET SHEET COUNT("ViewProArea";3)
```



Ver también

[VP Get sheet count](#)

## VP SET SHEET NAME

VP SET SHEET NAME ( *vpAreaName* : Text ; *name* : Text {; *index*: Integer} )

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro
<i>name</i>	Texto	->	Nuevo nombre para la hoja
<i>index</i>	Integer	->	Index of the sheet to be renamed

### Descripción

The `VP SET SHEET NAME` command renames a sheet in the document loaded in *vpAreaName*.

In *vpAreaName*, pass the name of the 4D View Pro area.

In *name*, pass a new name for the sheet.

In *index*, pass the index of the sheet to rename.

La indexación comienza en 0.

If no *index* is passed, the command renames the current sheet.

The new name cannot contain the following characters: `*`, `:`, `[`, `]`, `?,\,/`

El comando no hace nada si:

- el nuevo nombre contiene caracteres prohibidos
- el nuevo valor del nombre está en blanco
- el nuevo nombre ya existe
- el índice pasado no existe

### Ejemplo

Set the third sheet's name to "Total first quarter":

```
VP SET SHEET NAME("ViewProArea";"Total first quarter";2)
```



## VP SET SHEET OPTIONS

VP SET SHEET OPTIONS ( *vpAreaName* : Text; *sheetOptions* : Object { ; *sheet* : Integer} )

Parámetros	Tipo		Descripción
vpAreaName	Objeto	->	Nombre del área 4D View Pro
sheetOptions	Objeto	->	Opciones de la hoja a definir
sheet	Objeto	->	Índice de la hoja (hoja actual si se omite)

## Descripción

The `VP SET SHEET OPTIONS` command allows defining various sheet options of the `vpAreaName` area.

Pass the name of the 4D View Pro area in `vpAreaName`. If you pass a name that does not exist, an error is returned.

Pass an object containing definitions for the options to set in the `sheetOptions` parameter. To view the full list of the available options, see the [Sheet Options](#) paragraph.

In the optional `sheet` parameter, you can designate a specific spreadsheet (counting begins at 0). If omitted, the current spreadsheet is used by default. You can explicitly select the current spreadsheet with the following constant:

- `vk current sheet`

## Ejemplo 1

You want to protect all cells except the range C5:D10:

```
// Activate protection on the current sheet
var $options : Object

$options:=New object
$options.isProtected:=True
VP SET SHEET OPTIONS("ViewProArea";$options)

// mark cells C5:D10 as 'unlocked'
VP SET CELL STYLE(VP Cells("ViewProArea";2;4;2;6);New object("locked";False))
```

## Ejemplo 2

You need to protect your document while your users can resize rows and columns:

```
var $options : Object

$options:=New object
// Activate protection
$options.isProtected:=True
$options.protectionOptions:=New object
// Allow user to resize rows
$options.protectionOptions.allowResizeRows=True;
// Allow user to resize columns
$options.protectionOptions.allowResizeColumns=True;

// Apply protection on the current sheet
VP SET SHEET OPTIONS("ViewProArea";$options)
```

## Ejemplo 3

You want to customize the colors of your sheet tabs, frozen lines, grid lines, selection background and selection border:

```

var $options : Object

$options:=New object
// Customize color of Sheet 1 tab
$options.sheetTabColor:="Black"
$options.gridline:=New object("color";"Purple")
$options.selectionBackColor:="rgb(255,128,0,0.4)"
$options.selectionBorderColor:="Yellow"
$options.frozenlineColor:="Gold"

VP SET SHEET OPTIONS("ViewProArea";$options;0)

// Customize color of Sheet 2 tab
$options.sheetTabColor:="red"

VP SET SHEET OPTIONS("ViewProArea";$options;1)

// Customize color of Sheet 3 tab
$options.sheetTabColor:="blue"

VP SET SHEET OPTIONS("ViewProArea";$options;2)

```

Resultado:

	A	B	E	F	G	H	I	J	
1									
8									
9									
10									
11									
12									
--									

The screenshot shows a spreadsheet interface with three tabs at the bottom: "Sheet1" (black), "Sheet2" (red), and "Sheet3" (blue). The grid lines are visible, but the column headers (A, B, E, F, G, H, I, J) are not. A yellow selection box highlights a range from cell F9 to H11. Row numbers 1 through 12 are visible on the left.

Ejemplo 4

You want to hide the grid lines as well as the row and column headers.

```

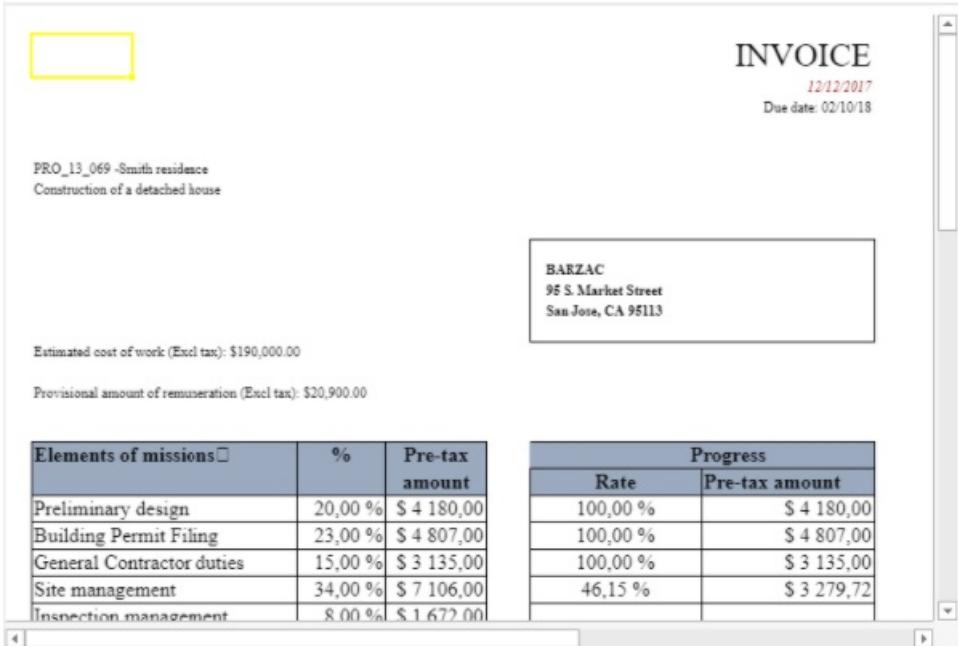
var $options : Object

$options:=New object
$options.gridline:=New object()
$options.gridline.showVerticalGridline:=False
$options.gridline.showHorizontalGridline:=False
$options.rowHeaderVisible:=False
$options.colHeaderVisible:=False

VP SET SHEET OPTIONS("ViewProArea";$options)

```

Resultado:



Ver también

[4D View Pro sheet options](#)

[VP Get sheet options](#)

## VP SET SHOW PRINT LINES

**VP SET SHOW PRINT LINES ( vpAreaName : Text {; visible : Boolean}{; index : Integer} )**

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
visible	Booleano	->	Print lines displayed if True (default), hidden if False
index	Integer	->	Índice de la hoja

### Descripción

The `VP SET SHOW PRINT LINES` command sets whether to display print preview lines in a spreadsheet..

In `vpAreaName`, pass the name of the 4D View Pro area.

In `visible`, pass `True` to display the print lines, and `False` to hide them. `True` se pasa por defecto.

In `index`, pass the index of the target sheet. If no index is specified, the command applies to the current sheet.

La indexación comienza en 0.

The position of a spreadsheet's print lines varies according to that spreadsheet's page breaks.

### Ejemplo

The following code displays print lines in a document's second sheet:

```
VP SET SHOW PRINT LINES("ViewProArea";True;1)
```

A screenshot of the Microsoft Excel interface. The ribbon at the top includes tabs for FILE, HOME, INSERT, FORMULAS, DATA, VIEW, and SETTINGS. The HOME tab is selected. The toolbar below the ribbon contains icons for Undo, Paste, Clipboard, Calibri font, font size 11, bold, italic, underline, alignment options (Wrap Text, Merge & Center), cell styles (General, %, Numbers), and editing tools (Cell Type, Styles, Cells, Editing). The worksheet area shows rows 1 through 21 and columns A through P. A red arrow points to the vertical scroll bar on the right side of the grid.

Con un salto de página:

A second screenshot of the Microsoft Excel interface, identical to the first one in layout and toolbars. The red arrow again points to the vertical scroll bar on the right side of the worksheet area, indicating a different scroll position or state compared to the first screenshot.

Ver también

[4D Get show print lines](#)

## VP SET TEXT VALUE

VP SET TEXT VALUE ( *rangeObj* : Object ; *textValue* : Text { ; *formatPattern* : Text } )

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto rango
textValue	Texto	->	Valor texto a definir
formatPattern	Texto	->	Formato del valor

## Descripción

The `VP SET TEXT VALUE` command assigns a specified text value to a designated cell range.

In *rangeObj*, pass a range of the cell(s) (created for example with `VP Cell` or `VP Column`) whose value you want to specify. If *rangeObj* includes multiple cells, the value specified will be repeated in each cell.

The *textValue* parameter specifies a text value to be assigned to the *rangeObj*.

The optional *formatPattern* defines a [pattern](#) for the *textValue* parameter.

## Ejemplo

```
VP SET TEXT VALUE(VP Cell("ViewProArea";3;2);"Test 4D View Pro")
```

## Ver también

[Cell Format](#)

[VP SET VALUE](#)

## VP SET TIME VALUE

`VP SET TIME VALUE ( rangeObj : Object ; timeValue : Text { ; formatPattern : Text } )`

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto rango
timeValue	Texto	->	Valor hora a definir
formatPattern	Texto	->	Formato del valor

## Descripción

The `VP SET TIME VALUE` command assigns a specified time value to a designated cell range.

In *rangeObj*, pass a range of the cell(s) (created for example with `VP Cell` or `VP Column`) whose value you want to specify. If *rangeObj* includes multiple cells, the value specified will be repeated in each cell.

The *timeValue* parameter specifies a time expressed in seconds to be assigned to the *rangeObj*.

The optional *formatPattern* defines a [pattern](#) for the *timeValue* parameter.

## Ejemplo

```
//Definir el valor para la hora actual
VP SET TIME VALUE(VP Cell("ViewProArea";5;2);Current time)

//Definir el valor para una hora específica con un formato designado
VP SET TIME VALUE(VP Cell("ViewProArea";5;2);?12:15:06?vk pattern long time)
```

## Ver también

[Cell Format](#)

[VP SET DATE TIME VALUE](#)  
[VP SET VALUE](#)

## VP SET VALUE

VP SET VALUE ( *rangeObj* : Object ; *valueObj* : Object )

Parámetros	Tipo		Descripción
<i>rangeObj</i>	Objeto	->	Objeto rango
<i>valueObj</i>	Objeto	->	Valores de la celda y opciones de formato

### Descripción

The `VP SET VALUE` command assigns a specified value to a designated cell range.

The command allows you to use a generic code to set and format the types of values in *rangeObj*, whereas other commands, such as `VP SET TEXT VALUE` and `VP SET NUM VALUE`, reduce the values to specific types.

In *rangeObj*, pass a range of the cell(s) (created for example with `VP Cell` or `VP Column`) whose value you want to specify. If *rangeObj* includes multiple cells, the value specified will be repeated in each cell.

The parameter *valueObj* is an object that includes properties for the value and the `format` to assign to *rangeObj*. Puede contener las siguientes propiedades:

Propiedad	Tipo	Descripción
<i>value</i>	Integer, Real, Boolean, Text, Date, Null	Value to assign to <i>rangeObj</i> (except- time). Pase null para borrar el contenido de la celda.
<i>time</i>	Real	Time value (in seconds) to assign to <i>rangeObj</i>
<i>format</i>	Texto	Patrón de propiedad valor/tiempo. For information on patterns and formatting characters, please refer to the <a href="#">Cell Format</a> paragraph.

### Ejemplo

```

//Set the cell value as False
VP SET VALUE(VP Cell("ViewProArea";3;2);New object("value";False))

//Set the cell value as 2
VP SET VALUE(VP Cell("ViewProArea";3;2);New object("value";2))

//Set the cell value as $125,571.35
VP SET VALUE(VP Cell("ViewProArea";3;2);New object("value";125571.35;"format";"$_(* #,##0.00_)"))

//Set the cell value as Hello World!
VP SET VALUE(VP Cell("ViewProArea";3;2);New object("value";"Hello World!"))

VP SET VALUE(VP Cell("ViewProArea";3;2);New object("value";"Hello World!"))

VP SET VALUE(VP Cell("ViewProArea";3;2);New object("value";"Hello World!"))

//Set the cell value as current date
VP SET VALUE(VP Cell("ViewProArea";4;2);New object("value";Current date))

//Set the cell value as current hour
VP SET VALUE(VP Cell("ViewProArea";5;2);New object("time";Current hour))

//Set the cell value as specific date and time
VP SET VALUE(VP Cell("ViewProArea";3;9);New object("value";!2024-12-18!); "time";?14:30:10?;"format";vk p)

//Erase cell content
VP SET VALUE(VP Cell("ViewProArea";3;9);New object("value";Null))

```

## Ver también

[Cell Format](#)  
[VP Get values](#)  
[VP SET VALUE](#)  
[VP SET BOOLEAN VALUE](#)  
[VP SET DATE TIME VALUE](#)  
[VP SET FIELD](#)  
[VP SET FORMULA](#)  
[VP SET NUM VALUE](#)  
[VP SET TEXT VALUE](#)  
[VP SET TIME VALUE](#)

## VP SET VALUES

`VP SET VALUES ( rangeObj : Object ; valuesCol : Collection )`

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto rango
valuesCol	Collection	->	Colección de valores

## Descripción

The `VP SET VALUES` command assigns a collection of values starting at the specified cell range.

In `rangeObj`, pass a range for the cell (created with `VP Cell`) whose value you want to specify. The cell defined in the `rangeObj` is used to determine the starting point.

- If `rangeObj` is not a cell range, only the first cell of the range is used.
- If `rangeObj` includes multiple ranges, only the first cell of the first range is used.

The `valuesCol` parameter is two-dimensional:

- The first-level collection contains subcollections of values. Cada subcolección define una línea. Each subcollection defines a row.
- Each subcollection defines cell values for the row. Values can be Integer, Real, Boolean, Text, Date, Null, or Object. If the value is an object, it can have the following properties:

Propiedad	Tipo	Descripción
value	Integer, Real, Boolean, Text, Date, Null	Valor de la celda (excepto - time)
time	Real	Valor hora (en segundos)

## Ejemplo

```
$param:=New collection
$param.push(New collection(1;2;3;False)) //first row, 4 values
$param.push(New collection) //second row, untouched
$param.push(New collection(4;5;Null;"hello";"world")) // third row, 5 values
$param.push(New collection(6;7;8;9)) // fourth row, 4 values
$param.push(New collection(Null;New object("value";Current date;"time";42))) //fifth row, 1 value

VP SET VALUES(VP Cell("ViewProArea";2;1);$param)
```

	A	B	C	D	E	F	G
1							
2			1	2	3	FALSE	
3							
4			4	5	hello	world	
5			6	7	8	9	
6			29/05/2019 0:00:42				
7							

## Ver también

[VP Get formulas](#)

[VP Get value](#)

[VP Get Values](#)

[VP SET FORMULAS](#)

[VP SET VALUE](#)

## VP SET WORKBOOK OPTIONS

`VP SET WORKBOOK OPTIONS ( vpAreaName : Text ; optionObj : Object)`

Parámetros	Tipo		Descripción
vpAreaName	Texto	->	Nombre de objeto formulario área 4D View Pro
optionObj	Objeto	->	Objeto que contiene las opciones del libro de trabajo a definir

## Descripción

`VP SET WORKBOOK OPTIONS` sets the workbook options in `vpAreaName`.

In `vpAreaName`, pass the name of the 4D View Pro area.

In `optionObj`, pass the workbook options to apply to `vpAreaName`.

If `optionObj` is empty, the command does nothing.

Modified workbook options are saved with the document.

The following table lists the available workbook options:

Propiedad	Tipo	Descripción												
<code>allowUserDragMerge</code>	booleano	The drag merge operation is allowed (select cells and drag the selection to merge cells)												
<code>allowAutoCreateHyperlink</code>	booleano	Enables automatic creation of hyperlinks in the spreadsheet.												
<code>allowContextMenu</code>	booleano	Se puede abrir el menú contextual integrado.												
<code>allowCopyPasteExcelStyle</code>	booleano	Styles from a spreadsheet can be copied and pasted to Excel, and vice-versa.												
<code>allowDynamicArray</code>	booleano	Permite arrays dinámicos en hojas de cálculo												
<code>allowExtendPasteRange</code>	booleano	Extends the pasted range if the pasted range is not enough for the pasted data												
<code>allowSheetReorder</code>	booleano	Se permite reordenar la hoja												
<code>allowUndo</code>	booleano	Deshacer ediciones está permitido.												
<code>allowUserDeselect</code>	booleano	Deselecting specific cells from a selection is allowed.												
<code>allowUserDragDrop</code>	booleano	Se permite arrastrar y soltar los datos del rango												
<code>allowUserDragFill</code>	booleano	Drag fill is allowed												
<code>allowUserEditFormula</code>	booleano	Formulas can be entered in cells												
<code>allowUserResize</code>	booleano	Columnas y filas redimensionables												
<code>allowUserZoom</code>	booleano	Se permite hacer zoom (ctrl + rueda del ratón)												
<code>autoFitType</code>	number	<p>Content is formatted to fit in cells, or cells and headers.</p> <p>Valores disponibles:</p> <table border="1"><thead><tr><th>Constante</th><th>Valor</th><th>Descripción</th></tr></thead><tbody><tr><td>vk auto fit type cell</td><td>0</td><td>El contenido se ajusta automáticamente a las celdas</td></tr><tr><td>vk auto fit type cell with header</td><td>1</td><td>El contenido se autoajusta a las celdas y encabezados</td></tr></tbody></table>	Constante	Valor	Descripción	vk auto fit type cell	0	El contenido se ajusta automáticamente a las celdas	vk auto fit type cell with header	1	El contenido se autoajusta a las celdas y encabezados			
Constante	Valor	Descripción												
vk auto fit type cell	0	El contenido se ajusta automáticamente a las celdas												
vk auto fit type cell with header	1	El contenido se autoajusta a las celdas y encabezados												
<code>backColor</code>	cadena	A color string used to represent the background color of the area, such as "red", "#FFFF00", "rgb(255,0,0)", "Accent 5". The initial backgroundcolor is hidden when a <code>backgroundImage</code> is set.												
<code>backgroundImage</code>	string / picture / file	Imagen de fondo para el área.												
<code>backgroundImageLayout</code>	number	<p>Cómo se muestra la imagen de fondo. Valores disponibles:</p> <table border="1"><thead><tr><th>Constante</th><th>Valor</th><th>Descripción</th></tr></thead><tbody><tr><td>vk image layout center</td><td>1</td><td>En el centro del área.</td></tr><tr><td>vk image layout none</td><td>3</td><td>In the upper left corner of the area with its original size.</td></tr><tr><td>vk image</td><td>0</td><td>Llena el área.</td></tr></tbody></table>	Constante	Valor	Descripción	vk image layout center	1	En el centro del área.	vk image layout none	3	In the upper left corner of the area with its original size.	vk image	0	Llena el área.
Constante	Valor	Descripción												
vk image layout center	1	En el centro del área.												
vk image layout none	3	In the upper left corner of the area with its original size.												
vk image	0	Llena el área.												

Propiedad	Tipo	Layout stretch			
		vk image layout zoom	2	Se muestra con su relación de aspecto original.	
calcOnDemand	booleano	Formulas are calculated only when they are demanded.			
columnResizeMode	number	Resize mode for columns. Valores disponibles:			
		Constante	Valor	Descripción	
		vk resize mode normal	0	Use normal resize mode (i.e remaining columns are affected)	
		vk resize mode split	1	Use split mode (i.e remaining columns are not affected)	
copyPasteHeaderOptions	number	Headers to include when data is copied to or pasted. Valores disponibles:			
		Constante	Valor	Descripción	
		vk copy paste header options all headers	3	Includes selected headers when data is copied; overwrites selected headers when data is pasted.	
		vk copy paste header options column headers	2	Includes selected column headers when data is copied; overwrites selected column headers when data is pasted.	
		vk copy paste header options no headers	0	Column and row headers are not included when data is copied; does not overwrite selected column or row headers when data is pasted.	
		vk copy paste header options row headers	1	Includes selected row headers when data is copied; overwrites selected row headers when data is pasted.	
customList	colección	The list for users to customize drag fill, prioritize matching this list in each fill. Cada elemento de colección es una colección de cadenas. See on <a href="#">GrapeCity's website</a> .			
cutCopyIndicatorBorderColor	cadena	Border color for the indicator displayed when the user cuts or copies the selection.			
cutCopyIndicatorVisible	booleano	Display an indicator when copying or cutting the selected item.			
defaultDragFillType	number	The default drag fill type. Valores disponibles:			
		Constante	Valor	Descripción	
		vk auto fill type auto	5	Llena automáticamente las celdas.	
		vk auto fill type clear values	4	Borra los valores de las celdas.	

Propiedad	Tipo	Descripción	fill type	0	Fills cells with all data objects, including values, formatting, and formulas.									
		copycells			objects, including values, formatting, and formulas.									
		vk auto fill type fill formatting only	2		Rellena las celdas sólo con formato.									
		vk auto fill type fill series	1		Fills cells with series.									
		vk auto fill type fill without formatting	3		Llena las celdas con valores y no con formato.									
enableAccessibility	booleano	Accessibility support is enabled in the spreadsheet.												
enableFormulaTextbox	booleano	The formula text box is enabled.												
grayAreaBackColor	cadena	A color string used to represent the background color of the gray area , such as "red", "#FFFF00", "rgb(255,0,0)", "Accent 5", and so on.												
highlightInvalidData	booleano	Los datos inválidos son resaltados.												
iterativeCalculation	booleano	Activa el cálculo iterativo. See on <a href="#">Grapecity's website</a> .												
iterativeCalculationMaximumChange	numeric	Maximum amount of change between two calculation values.												
iterativeCalculationMaximumIterations	numeric	Número de veces que la fórmula debe recalcular.												
newTabVisible	booleano	Display a special tab to let users insert new sheets.												
numbersFitMode	number	Changes display mode when date/number data width is longer than column width. Valores disponibles:												
		<table border="1"> <thead> <tr> <th>Constante</th><th>Valor</th><th>Descripción</th></tr> </thead> <tbody> <tr> <td>vk numbers fit mode mask</td><td>0</td><td>Replace data content with "###" and shows tip</td></tr> <tr> <td>vk numbers fit mode overflow</td><td>1</td><td>Display data content as a string. If next cell is empty, overflow the content.</td></tr> </tbody> </table>				Constante	Valor	Descripción	vk numbers fit mode mask	0	Replace data content with "###" and shows tip	vk numbers fit mode overflow	1	Display data content as a string. If next cell is empty, overflow the content.
Constante	Valor	Descripción												
vk numbers fit mode mask	0	Replace data content with "###" and shows tip												
vk numbers fit mode overflow	1	Display data content as a string. If next cell is empty, overflow the content.												
pasteSkipInvisibleRange	booleano	Pegar u omitir pegar datos en rangos invisibles: <ul style="list-style-type: none"> <li>• False (por defecto): pegar datos</li> <li>• True: omitir el pegado en rangos invisibles</li> </ul> See <a href="#">Grapecity's docs</a> for more information on invisible ranges.												
referenceStyle	number	Style for cell and range references in cell formulas. Valores disponibles:												
		<table border="1"> <thead> <tr> <th>Constante</th><th>Valor</th><th>Descripción</th></tr> </thead> <tbody> <tr> <td>vk reference style A1</td><td>0</td><td>Utilizar el estilo A1.</td></tr> <tr> <td>vk reference style R1C1</td><td>1</td><td>Utilizar el estilo R1C1</td></tr> </tbody> </table>				Constante	Valor	Descripción	vk reference style A1	0	Utilizar el estilo A1.	vk reference style R1C1	1	Utilizar el estilo R1C1
Constante	Valor	Descripción												
vk reference style A1	0	Utilizar el estilo A1.												
vk reference style R1C1	1	Utilizar el estilo R1C1												
resizeZeroIndicator	number	Drawing policy when the row or column is resized to zero. Valores disponibles:												
		<table border="1"> <thead> <tr> <th>Constante</th><th>Valor</th><th>Descripción</th></tr> </thead> <tbody> <tr> <td>vk resize zero indicator default</td><td>0</td><td>Uses the current drawing policy when the row or column is resized to zero.</td></tr> </tbody> </table>				Constante	Valor	Descripción	vk resize zero indicator default	0	Uses the current drawing policy when the row or column is resized to zero.			
Constante	Valor	Descripción												
vk resize zero indicator default	0	Uses the current drawing policy when the row or column is resized to zero.												

Propiedad	Tipo	Descripción	1	Draws two short lines when the row or column is resized to zero.
rowResizeMode	number	La forma en que se redimensionan las líneas. Los valores disponibles son los mismos qe columnResizeMode		
scrollbarAppearance	number	Apariencia de la barra de desplazamiento. Valores disponibles:		
		Constante	Valor	Descripción
		vk scrollbar appearance mobile	1	Apariencia de la barra de desplazamiento móvil.
		vk scrollbar appearance skin (por defecto)	0	Aspecto de la barra de desplazamiento clásica similar a la de Excel.
scrollbarMaxAlign	booleano	The scroll bar aligns with the last row and column of the active sheet.		
scrollbarShowMax	booleano	The displayed scroll bars are based on the entire number of columns and rows in the sheet.		
scrollByPixel	booleano	Activar desplazamiento de precisión por píxel.		
scrollIgnoreHidden	booleano	La barra de desplazamiento ignora líneas o columnas ocultas.		
scrollPixel	integer	Decides scrolling by that number of pixels at a time when scrollByPixel is true. The final scrolling pixels are the result of scrolling delta * scrollPixel . For example: scrolling delta is 3, scrollPixel is 5, the final scrolling pixels are 15.		
showDragDropTip	booleano	Display the drag-drop tip.		
showDragFillSmartTag	booleano	Display the drag fill dialog.		
showDragFillTip	booleano	Display the drag-fill tip.		
showHorizontalScrollbar	booleano	Mostrar la barra de desplazamiento horizontal.		
showResizeTip	number	How to display the resize tip. Valores disponibles:		
		Constante	Valor	Descripción
		vk show resize tip both	3	Horizontal and vertical resize tips are displayed.
		vk show resize tip column	1	Only the horizontal resize tip is displayed.
		vk show resize tip none	0	No resize tip is displayed.
		vk show resize tip row	2	Only the vertical resize tip is displayed.
showScrollTip	number	How to display the scroll tip. Valores disponibles:		
		Constante	Valor	Descripción
		vk show scroll tip both	3	Se muestran las extremidades horizontales y verticales.
		vk show scroll tip horizontal	1	Only the horizontal scroll tip is displayed.

Propiedad	Tipo	Descripción		
		vk show scroll tip none	No scroll tip is displayed.	
		vk show scroll tip vertical	2	Only the vertical scroll tip is displayed.
showVerticalScrollbar	booleano	Display the vertical scroll bar.		
tabEditable	booleano	The sheet tab strip can be edited.		
tabNavigationVisible	booleano	Display the sheet tab navigation.		
tabStripPosition	number	Position of the tab strip. Valores disponibles:		
		Constante	Valor	Descripción
		vk tab strip position bottom	0	Tab strip position is relative to the bottom of the workbook.
		vk tab strip position left	2	Tab strip position is relative to the left of the workbook.
		vk tab strip position right	3	Tab strip position is relative to the right of the workbook.
		vk tab strip position top	1	Tab strip position is relative to the top of the workbook.
tabStripRatio	number	Percentage value (0.x) that specifies how much of the horizontal space will be allocated to the tab strip. The rest of the horizontal area (1 - 0.x) will be allocated to the horizontal scrollbar.		
tabStripVisible	booleano	Display the sheet tab strip.		
tabStripWidth	number	Width of the tab strip when position is left or right. Por defecto y el mínimo es 80.		
useTouchLayout	booleano	Whether to use touch layout to present the Spread component.		

## Ejemplo

To set the allowExtendPasteRange option in "ViewProArea":

```
var $workbookOptions : Object
$workbookOptions:= New Object
$workbookOptions.allowExtendPasteRange:=True
VP SET WORKBOOK OPTIONS("ViewProArea";$workbookOptions)
```

## Ver también

[VP Get workbook options](#)

## VP SHOW CELL

VP SHOW CELL ( *rangeObj* : Object { ; *vPos* : Integer; *hPos* : Integer } )

Parámetros	Tipo		Descripción
rangeObj	Objeto	->	Objeto rango
vPos	Integer	->	Posición vertical de la vista de la celda o de la línea
hPos	Integer	->	Posición horizontal de la vista de la celda o de la línea

## Descripción

The `VP SHOW CELL` command vertically and horizontally repositions the view of the *rangeObj*.

In *rangeObj*, pass a range of cells as an object to designate the cells to be viewed. The view of the *rangeObj* will be positioned vertically or horizontally (i.e., where *rangeObj* appears) based on the *vPos* and *hPos* parameters. The *vPos* parameter defines the desired vertical position to display the *rangeObj*, and the *hPos* parameter defines the desired horizontal position to display the *rangeObj*.

Los siguientes selectores están disponibles:

Selector	Descripción	Disponible con <i>vPos</i>	Disponible con <i>hPos</i>
<code>vk position bottom</code>	Alineación vertical a la parte inferior de la celda o de la línea.	X	
<code>vk position center</code>	Alineación al centro. The alignment will be to the cell, row, or column limit according to the view position indicated: <ul style="list-style-type: none"> <li>• Posición vertical de la vista - celda o línea</li> <li>• Posición horizontal de la vista - celda o columna</li> </ul>	X	X
<code>vk position left</code>	Horizontal alignment to the left of the cell or column		X
<code>vk position nearest</code>	Alignment to the closest limit (top, bottom, left, right, center). The alignment will be to the cell, row, or column limit according to the view position indicated: <ul style="list-style-type: none"> <li>• Vertical view position (top, center, bottom) - cell or row</li> <li>• Horizontal view position (left, center, right) - cell or column</li> </ul>	X	X
<code>vk position right</code>	Horizontal alignment to the right of the cell or column		X
<code>vk position top</code>	Alineación vertical a la parte superior de la celda o de la línea	X	

This command is only effective if repositioning the view is possible. For example, if the *rangeObj* is in cell A1 (the first column and the first row) of the current sheet, repositioning the view will make no difference because the vertical and horizontal limits have already been reached (i.e., it is not possible to scroll any higher or any more to the left). The same is true if *rangeObj* is in cell C3 and the view is repositioned to the center or the bottom right. La vista permanece inalterada.

## Ejemplo

You want to view the cell in column AY, row 51 in the center of the 4D View Pro area:

```

$displayCell:=VP Cell("myVPArea";50;50)
// Mover la vista para mostrar la celda
VP SHOW CELL($displayCell;vk position center;vk position center)

```

Resultado:

	AV	AW	AX	AY	AZ	BA	BB	BC
42								
43								
44								
45								
46								
47								
48								
49								
50								
51					Hello World			
52								
53								
54								
55								
56								
57								
58								
59								
60								

The same code with the vertical and horizontal selectors changed to show the same cell positioned at the top right of the 4D View Pro area:

```

$displayCell:=VP Cell("myVPArea";50;50)
// Mover la vista para mostrar la celda
VP SHOW CELL($displayCell;vk position top;vk position right)

```

Resultado:

	AS	AT	AU	AV	AW	AX	AY	AZ
51							Hello World	
52								
53								
54								
55								
56								
57								
58								
59								
60								
61								
62								
63								
64								
65								
66								
67								
68								
69								
70								

Ver también

[VP ADD CELL](#)  
[VP Get active cell](#)  
[VP Get selection](#)  
[VP RESET SELECTION](#)  
[VP SET ACTIVE CELL](#)  
[VP SET SELECTION](#)

## VP SUSPEND COMPUTING

VP SUSPEND COMPUTING ( *vpAreaName* : Text )

Parámetros	Tipo		Descripción
<i>vpAreaName</i>	Texto	->	Nombre de objeto formulario área 4D View Pro

### Descripción

The `VP SUSPEND COMPUTING` command stops the calculation of all formulas in *vpAreaName*. This command is useful when you want to suspend calculations in this 4D View Pro area so you can manually make modifications to formulas without encountering errors before you've finished making the changes.

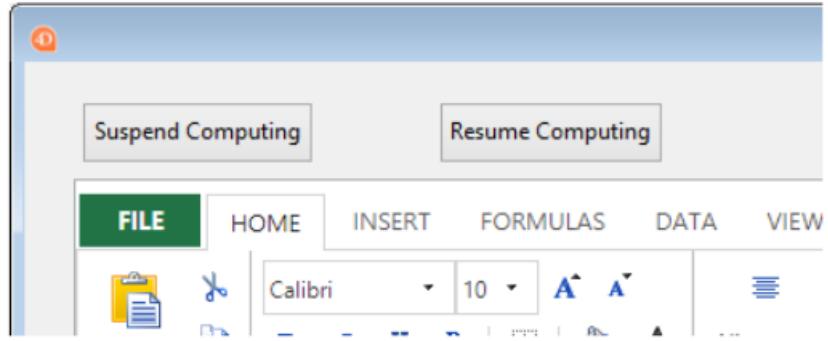
The command pauses the calculation service in 4D View Pro. Formulas that have already been calculated remain unchanged, however any formulas added after `VP SUSPEND COMPUTING` command is executed are not calculated.

In *vpAreaName*, pass the name of the 4D View Pro area. If you pass a name that does not exist, an error is returned.

The 4D View Pro calculation service maintains a counter of suspend/resume actions. Therefore, each execution of `VP SUSPEND COMPUTING` command must be balanced by a corresponding execution of the `VP RESUME COMPUTING` command. Any formula impacted by modifications made while calculations are suspended will be recalculated when the command is executed.

### Ejemplo

You've added two buttons to the form so that the user can suspend/resume calculations:



El código del botón Suspend Computing:

```
//pause calculations while users enter information
If(FORM Event.code=On Clicked)

    VP SUSPEND COMPUTING("ViewProArea")

End if
```

```
If(FORM Event.code=On Clicked)

    VP RESUME COMPUTING("ViewProArea")

End if
```

Ver también

[VP RECOMUTE FORMULAS](#)  
[VP RESUME COMPUTING](#)

# Programación avanzada con Javascript

A 4D View Pro Area is a [Web Area form object](#) that uses the [embedded web rendering engine](#). As such, it behaves just like any other web area, and you can get it to execute Javascript code by calling the [WA Evaluate Javascript](#) 4D command.

Since 4D View Pro is powered by the [SpreadJS spreadsheet solution](#), you can also call SpreadJS Javascript methods in 4D View Pro areas.

## Ejemplo: ocultar la cinta

Since 4D View Pro is a web area, you can select a webpage element and modify its behavior using Javascript. The following example hides the spreadJS [Ribbon](#):

```
//Button's object method

var $js; $answer : Text

$js:="document.getElementsByClassName('ribbon')[0].setAttribute('style','d
$js+="window.dispatchEvent(new Event('resize'));""

$answer:=WA Evaluate JavaScript(*; "ViewProArea"; $js)
```

## Llamar los métodos Javascript de SpreadJS

You can tap into the SpreadJS library of Javascript methods and call them directly to control your spreadsheets.

4D has a built-in [Utils.spread](#) expression that points at the spreadsheet (also called workbook) inside the 4D View Pro area, making it simpler to call the SpreadJS [Workbook methods](#).

### Ejemplo

The following code undoes the last action in the spreadsheet:

```
WA Evaluate JavaScript(*; "ViewProArea"; "Utils.spread.undoManager().undo()
```

## Repositorio 4D View Pro Tips

[4D-View-Pro-Tips](#) is a GitHub repository that contains a project full of useful functions, allowing to manage floating pictures, sort columns or rows, create a custom culture, and much more! Feel free to

clone it and experiment with the project.