4D Open for 4th Dimension®

for Windows® and Mac™OS

4D Open for 4th Dimension
by
4D S.A.

4D Open for 4th Dimension
Version 200x for Windows® and Mac™ OS

IMPORTANT LICENSE INFORMATION

Use of this Software is subject to the 4D Product Line License Agreement, which is provided in electronic form with the Software. Please read the 4D Product Line License Agreement carefully before completely installing or using the Software.

Contents

1

Introduction

What is 4D Open for 4D?

4D Open for 4D (4D Open for 4th Dimension) is a 4D connectivity plug-in that enables you to simultaneously connect to multiple 4D Servers from 4th Dimension (the single-user version of 4D), from 4D Client (in addition to its normal connection capability) or from 4D Server itself (using version 6 stored procedures and triggers).

From either of these applications you can:

- Get a description of the database structures served the 4D Server,
- Get information from 4D Server (such as the application version, the connected users, the processes currently running on the server machine),
- Query and retrieve data from 4D Server, and
- Update and modify that data.

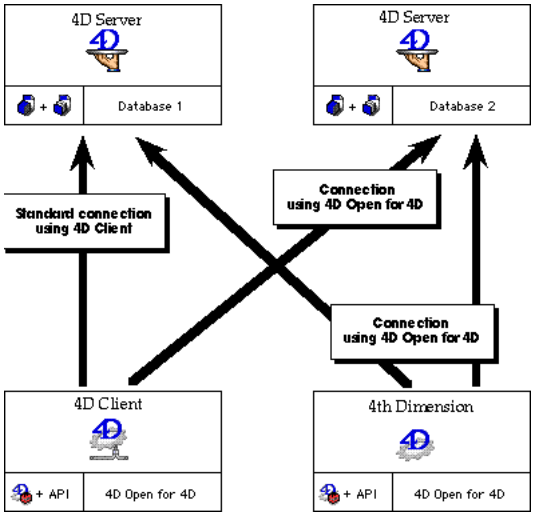
4D Open for 4D adds new commands to 4th Dimension. All 4D Open for 4D routine names are prefixed by the letters “OP” to distinguish them from standard 4D commands and from commands added by other plug-ins.

What is 4D Server?

4D Server is the Client/Server multi-user version of 4th Dimension from 4D, Inc. With 4D Server, you can create and use multi-user databases and custom applications in a client/server architecture.

The 4D Server application itself is divided into two parts: 4D Server and 4D Client. 4D Server manages the database and responds to user requests. Using 4D Client, users can access the 4D Server database from their workstations and perform database operations such as adding data, generating reports and modifying the database design. Anything that can be done with 4th Dimension can be done using 4D Server and 4D Client.

4D Open for 4th Dimension routines are called from 4th Dimension or 4D Client. The following diagram shows how 4D Client and 4th Dimension can access 4D Server databases:



4D Open for 4th Dimension enables you to establish multiple connections to a single 4D Server or to connect to multiple 4D Servers simultaneously from 4th Dimension or 4D Client.

4D Open for 4th Dimension is a set of external functions. It allows you to perform the following database operations:

Connect to a 4D Server Database

You can connect to one or more 4D Server databases simultaneously. During this stage, you initialize the client application and workstation and select the database. You can specify the database name procedurally or you can allow the user to select a database using a dialog box.

Access the Structure Definition

You can retrieve the names of the tables and fields and the definitions of fields. Using these routines, you can easily implement a user interface that displays the database structure (for instance, a pop-up menu listing the tables).

Execute Queries

When you want to update data, you generally select all the data in a table or perform a search to find specific data and create a selection of records. Users can then add, update or delete data in this selection.

There is only one selection of records at a time for each table in each process. This selection of records remains valid until it is changed (e.g., another query is processed).

Use Named Selections

4D Open for 4th Dimension supports named selections, allowing optimum use of your queries and selections. After performing a query or search, you can have 4D Server maintain the resulting selection of records as a named selection. Using this named selection, you can instantaneously retrieve the selection without re-executing the query.

Use Arrays

4D Open for 4th Dimension provides routines for sending and retrieving data using arrays that behave like data buffers.

Manage Multiple Users

You can manage database operations in a multi-user environment by setting the access modes of tables to read/write or read-only, and unlocking records when you are finished with them.

Use Transactions

You can perform complex updates, including those involving multiple tables, while maintaining data integrity. To provide this functionality, 4D Open for 4th Dimension includes routines for starting, validating and cancelling transactions.

Perform Calculations

4D Open for 4th Dimension can help you quickly summarize a set of data by performing numerical calculations, such as the sum, average, maximum and minimum.

Data Security

Clients accessing 4D Server through 4D Open for 4th Dimension are subject to the 4th Dimension password access system. Such clients must provide valid user names and passwords if the database is password protected. Access privileges to records for loading, writing, updating and deleting are also required.

All 4D Open for 4th Dimension (4D Open for 4D) plug-in routines are functions. Each function returns an error code result that tells you whether or not the operation was successfully performed. If the function returns 0, no error has occurred.

4D Open for 4D functions should always be assigned a Long Integer (Longint) variable that will receive the error code. The following code shows you how to use a 4D Open for 4D function:

```
   ` Do something and get error code result
$errCode:=OP Set option (3;1)
```

Some 4D Open for 4D routines such as OP Get field properties return values as well as error codes. The values are returned in the parameters, and the error code is returned in the Longint variable assigned to the routine. In the following line of code, the error code is returned in $errCode, and the values returned by the routine are placed into the parameters (which have been previously declared). The connID and tabID parameters are passed to this function to retrieve information about them in the other parameters:

```
   $errCode:=OP Get field properties
(connID;tabID;tabInv;fNames;fTypes;fLen;fIndex;fInv)
```

Nature of the parameters passed to 4D Open for 4D routines

The types of parameters passed to a routine depend on whether the routine returns a value in the parameter and whether or not the parameter refers to a 4th Dimension object.

Parameters not returning a value

You can pass an expression, a field, a variable (local, process, or interprocess) or an array (process or interprocess).

You cannot pass local arrays.

Since you are working in a multi-process environment, if you pass interprocess variables or arrays, be sure to protect the use of these variables and arrays by using local semaphores.

Examples:

```
$ErrCode:=OP Enter password (arUser;$UserName;$Password)
$ErrCode:=OP Enter password (◊arUser;$UserName;$Password)
```

Parameters returning a value

You can pass a field, a variable (local, process, or interprocess) or an array (process or interprocess).

You cannot pass local arrays.

Since you are working in a multi-process environment, if you pass interprocess variables or arrays, be sure to protect the use of these variables and arrays by using local semaphores.

Examples:

```
$ErrCode:=OP Count tables ($ConnID;[Servers]NbFiles)
$ErrCode:=OP Count tables ($ConnID;$NbFiles)
$ErrCode:=OP Count tables ($ConnID;NbFiles)
$ErrCode:=OP Count tables ($ConnID;◊NbFiles)
$ErrCode:=OP Get all tablenames ($ConnID;arFName)
$ErrCode:=OP Get all tablenames ($ConnID;◊arFName)
```

Parameters referring to a 4D object (fields, variables, arrays)

You can pass a reference to a field (pointer or number) or to a process or interprocess variable or array (pointer or name).

You cannot pass a pointer to an array element.

You cannot pass a reference to a local variable or array.

Since you are working in a multi-process environment, if you pass interprocess variables or arrays, be sure to protect the use of these variables and arrays by using local semaphores.

Note: You can get/set BLOB fields in the same way you do it with other fields.

Examples:

```
$ErrCode:=OP Single query ($ConnID;$FileID;$FieldID;"=";»[Data]Value;$NbFound)
$ErrCode:=OP Single query ($ConnID;$FileID;$FieldID;"=";»vValue;$NbFound)
$ErrCode:=OP Single query ($ConnID;$FileID;$FieldID;"=";»◊vValue;$NbFo und)
$ErrCode:=OP Define bind by numbers ($BindID;10;4;2;4;"")
$ErrCode:=OP Define bind by numbers ($BindID;10;4;0;0;"vName")
$ErrCode:=OP Define bind by numbers ($BindID;10;4;0;0;"◊vName")
$ErrCode:=OP Define bind by numbers ($BindID;10;4;0;0;"arValue")
$ErrCode:=OP Define bind by numbers ($BindID;10;4;0;0;"◊arValue")
$ErrCode:=OP Define bind by pointer ($BindID;10;4;»vName)
$ErrCode:=OP Define bind by pointer ($BindID;10;4;»◊vName)
$ErrCode:=OP Define bind by pointer ($BindID;10;4;»arValue)
$ErrCode:=OP Define bind by pointer ($BindID;10;4;»◊arValue)
```

What you should know

To use 4D Open for 4D, you should be familiar with the 4th Dimension programming language.

What you need

To use 4D Open for 4D, you must have 4D Server and either 4D Client or 4th Dimension. All version numbers must match; for example with 4D Open 4D 2003 you must use 4D Server 2003 as well as 4D Client or 4th Dimension 2003. You cannot mix different versions.

This manual describes the 4D Open for 4th Dimension routines. It assumes that you are familiar with the 4th Dimension programming language. For more information about the 4th Dimension programming language, refer to the *4th Dimension Language Reference*.

Starting with Chapter 2, the chapters in this manual correspond to the function categories (themes) in the 4th Dimension Method editor.

Conventions

This manual uses certain conventions to help you understand the material.

In the syntax descriptions, the -> and <- symbols mean the following:

Symbol	Description
->	Parameter is only passed, so you can pass a value, a variable, or an array
<-	Parameter is returned, so you must pass a variable or an array. The array can be process or interprocess, but not local.

This manual uses a number of specific conventions similar to those in the 4th Dimension Method editor to identify procedure code and commands.

4D code examples appear in a special font. For example:

```
ALL RECORDS
```

4D Open for 4th Dimension external functions appear in bold-italic. For example:

OP All records

Parameters of these external functions appear in a different typeface. For example:

ConnectionID

About the Examples

The examples in this manual illustrate the use of the 4D Open for 4th Dimension functions. These examples may not have been optimized. In addition, be sure to use error-checking when using them in your code.

2

Network Components

Starting with 4D Open for 4th Dimension version 2003, you access 4D Server databases using TCP/IP network protocol only. This protocol is automatically loaded; no additional settings are required.

OP Count network components

Network Components

version 2003 (Modified)

Compatibility Note: Starting with version 2003, 4D Open 4D only uses TCP/IP protocol. This command is maintained only for compatibility reasons and should no longer be used.

OP Count network components (nbOfNetComp) → Longint

Parameter	Type		Description
nbOfNetComp	Longint	←	Number of installed network components
Function result	Longint	←	Error code result for the function

OP Get network component info

Network Components

version 2003 (Modified)

Compatibility Note: Starting with version 2003, 4D Open 4D only uses TCP/IP protocol. This command is maintained only for compatibility reasons and should no longer be used.

OP Get network component info (netCompNumber; netCompID; netCompName) → Longint

Parameter	Type		Description
netCompNumber	Integer	→	Network component number
netCompID	Integer	←	Reference number of specified network component
netCompName	String	←	Name of specified network component
Function result	Longint	←	Error code result for the function

Deprecated: This command is disabled in current versions of 4D Open for 4th Dimension. The network component (TCP/IP) is now loaded automatically.

Deprecated: This command is no longer useful and is disabled in current versions of 4D Open for 4th Dimension.

OP Request (connectID; status) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
status	Longint	←	Is request in progress?
Function result	Longint	←	Error code

Description

When running in asynchronous mode, the command OP Request allows you to test if the running request has completed.

4D Open for 4D cannot execute more than one request at a time. OP Request returns 1 in the status parameter if the request has completed, otherwise it returns 0 (zero).

In previous versions of 4D Open for 4D, it was not possible to test if a request was completed in asynchronous mode — actually the only solution was to try to start a new one (-32003;-32001;10154).

See Also

OP Open connection.

3

Network Utilities

The following routines are described in this section:

• OP Get station name - obtains the name of the machine on which you are working.
• OP Start remote connection - initiates a new remote connection using ARA on Macintosh or RAS on Windows NT or Windows 95.

• OP End Remote connection - terminates a remote connection.

• OP Remote Connection Status - provides a means to check the status of a connection attempt.

OP Get station name → String

Parameter	Type		Description
This command does not require any parameters			
Function result	String	←	Workstation name

Description

OP Get station name returns the name of the workstation on which the application is running.

On a Macintosh, this name is set in the Sharing Setup Control Panel.
On a computer running Windows, this name is set in the Network Control Panel

Example

```
C_LONGINT ($errCode;netCompID;$servID; $connID)
C_STRING (32;$WS)

$errCode := LoadNetComp ("TCP/IP";->netCompID)
If ($errCode = 0)
      ` Turn the adress string into a server entry
   $errCode := OP Find 4D Server (netCompID;"accounting.4D.com";$servID)
End if
If ($errCode =0)
      ` Get workstation name
⇒     $WS := OP Get station name
      ` Open connection with the server
   $errCode := OP Open connection ($servID; $connID; $WS; "Jerry"; "YRREJ";
                                                                  "DevSearch")
End if
```

OP Start Remote Connection (path; user; password; async) → Longint

Parameter	Type		Description
path	String	→	Path
user	String	→	User Name
password	String	→	Password
async	Boolean	→	Initiate connection asyncronously
Function result	Longint	←	Error code result for the function

Description

OP Start Remote Connection initiates a new remote connection using ARA on Macintosh or RAS on Windows NT or Windows 95.

The path parameter is the pathname to an ARA client document when using ARA, or a phonebook entry when using RAS. If the path parameter is passed as an empty string, a getfile prompts the user for the file (Macintosh). The user parameter is not implemented on ARA at this time. If the password parameter is passed as an empty string, OP Start Remote Connection uses the password stored in the file (if any).

Examples

Asynchronous:

```
    C_LONGINT(status1;$ErrCode;error)
    C_STRING(100;devType;devName)
    C_TEXT($errorText)
⇒   $ErrCode:=OP Start Remote Connection("Home";"MyName";"MyPassword";True)
    Repeat
       $ErrCode:=OP Remote Connection Status(status1;error;devType;devName)
    Until(status#1)
    $errorText:=OP Get Error Text($ErrCode)
    ...
    $errorText:=OP End Remote Connection()
```

Syncronous:

```
⇒   $ErrCode:=OP Start Remote Connection("Home";"MyName";"MyPassword";False)
          `yields control back once connected or if error has occured
```

See Also

OP End Remote Connection, OP Remote Connection Status.

OP End Remote Connection → Longint

Parameter	Type		Description
This command does not require any parameters			
Function result	Longint	←	Error code result for the function

Description

OP End Remote Connection terminates a remote connection that has been succesfully initiated using OP Start Remote Connection. On Macintosh, OP End Remote Connection function returns the ARA **ioResultCode** into the error result code. On Windows, it returns the RAS **ErrorCode** into the error result code.

Example

```
C_LONGINT(status1;$ErrCode;error)
C_STRING(100;devType;devName)
C_TEXT($errorText)

$ErrCode:=OP Start Remote Connection("Home";"MyName";"MyPassword";True)

Repeat
   $ErrCode:=OP Remote Connection Status(status1;error;devType;devName)
Until(status#1)

$errorText:=OP Get Error Text($ErrCode)
...
⇒  $errorText:=OP End Remote Connection()
```

See Also

OP Remote Connection Status, OP Start Remote Connection.

OP Remote Connection Status (phase; error; connectedTo; lastMessage) → Longint

Parameter	Type		Description
phase	Longint	←	Status of remote connection
error	Longint	←	Error
connectedTo	String	←	Connected to
lastMessage	String	←	Last message
Function result	Longint	←	Error code result for the function

Description

OP Remote Connection Status provides a means to check the status of a connection attempt that was initiated using OP Start Remote Connection. After a connection has been succesfully initiated, the phase parameter returns a value of one (1).

The possible values for the phase parameter are:
0: Connected
1: Connection in progress
2: Disconnected

Example

```
C_LONGINT(status1;$ErrCode;error)
C_STRING(100;devType;devName)
C_TEXT($errorText)

$ErrCode:=OP Start Remote Connection("Home";"MyName";"MyPassword";True)

Repeat
⇒    $ErrCode:=OP Remote Connection Status(status1;error;devType;devName)
Until(status#1)

$errorText:=OP Get Error Text($ErrCode)
...
$errorText:=OP End Remote Connection()
```

See Also

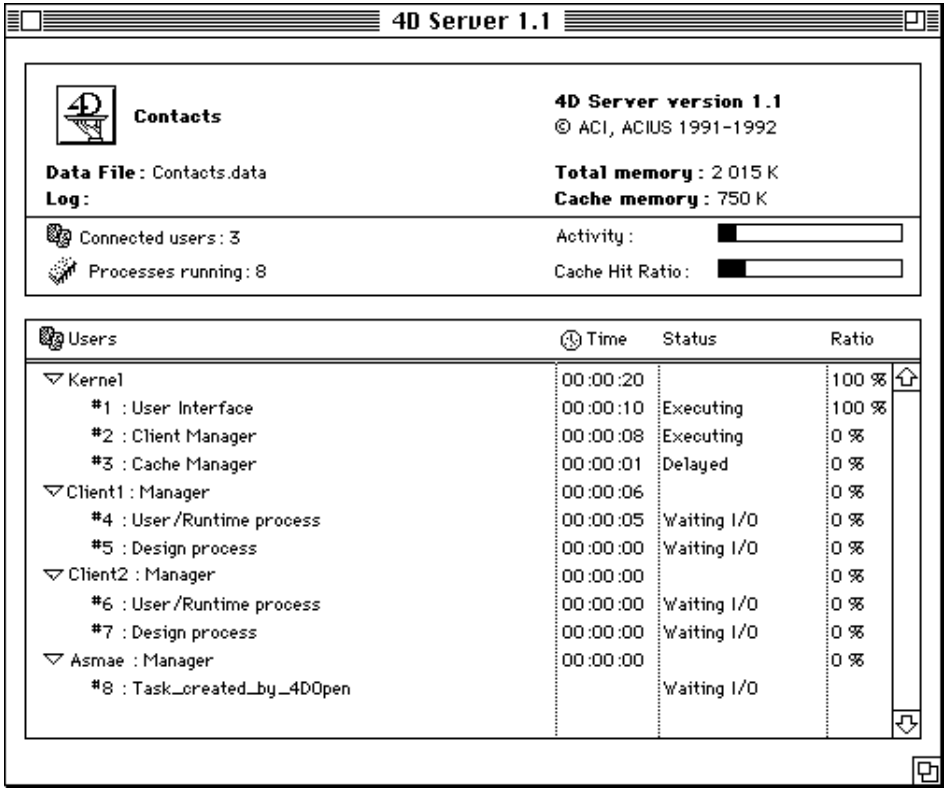
OP End Remote Connection, OP Start Remote Connection.

4

Connections

Once a 4D Server database has been identified over the network, you can open a connection to it by using OP Open connection. Remember that you can open multiple connections to the same database and can open multiple databases simultaneously.

For each connection opened using OP Open connection, a new process is created on the target server. If you are using a color monitor, the new process appears in green in the 4D Server window to distinguish it from 4D Client connections. The 4D Server window is shown here:



For each user, the 4D Server window displays the name of the client station, the name of the user, and the names of the user's processes. You supply this information to OP Open connection.

Processes

The first time that you open a connection to a database from a 4D Open for 4th Dimension client, the client is registered as being connected to the database and a new process is started for the connection. If you connect to the database from the same client again, another process is added for the client. By connecting to a database several times, you can have multiple processes running under the same client.

In 4D Server, processes behave like separate connections to the same database. You may want to open several processes for the same client so that you can perform different operations in each one. For example, you might want to retrieve and display a list of records in one process and allow the user to add records in another process.

Closing a Connection

You can close a connection by using OP Close connection.

Providing Access to Non-4D Clients

In 4th Dimension and 4D Client, you can allow non-4D Client applications to access a database. A non-4D Client application can be any application that uses 4D Open to connect to a 4D Server database.

To allow non-4D Client applications to connect to a 4D Server database, you must deselect the "Allow 4D Client connections only" check box in the Database properties dialog box. This feature was added in 4D Server version 1.1.

If this check box has already been selected for a database, you will not be able to connect to the database from 4D Open for 4th Dimension. If you attempt to connect to the database, an error will be returned:

- If you are using 4D Server version 1.1, error -9956 is returned.
- If you are using 4D Server version 1.1.1 or greater, error -9947 is returned.
- If you are using 4D Server version 1.1.1, you will also receive an error if the user connecting to the database is not part of the password group designated to have 4D Open access to the database.

Providing Access Privileges to a 4D Open Group

In 4D Server version 1.1.1 or greater, the database designer can give special access to a group of users so that they can connect to the database via 4D Open.

Remember that the "Allow 4D Client connections only" check box must also be deselected to give access to users who are using 4D Open to connect to the 4D Server database. This group can be selected from the 4D Open access pop-up menu. If no groups have been created, "All Groups" is chosen by default. If one or more groups have been created, no group is selected by default. If a user is not in the group selected from the 4D Open access pop-up menu, error -9944 appears when he/she tries to connect to the database via 4D Open.

To display a connection dialog box, call OP Select 4D Server. In this case, the user can choose among a list of databases opened by 4D Server.

To find a 4D Server database on the server without the user having to select one, you can call OP Find 4D Server.

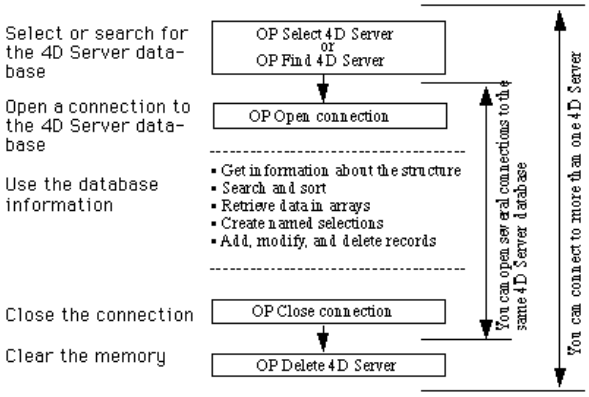
Once you have selected a 4D Server database you can open a connection to it by calling OP Open connection.

To close the connection opened with OP Open connection, call OP Close connection.

To clear the space in memory allocated (by OP Select 4D Server or OP Find 4D Server) for keeping track of the server, call OP Delete 4D Server.

The routines described in this chapter enable you to select a 4D Server database as well as to open and close a connection to the selected database.

The following diagram depicts how a connection to a 4D Server database is opened from 4th Dimension or 4D Client:



Using Multiple Connections and Databases

Using the 4D Open routines, a single client can open multiple connections. The client can connect to several databases at the same time. Each connection corresponds to a different 4D Server process.

With 4D Open, each user can open six concurrent connections (processes) to the same server. After the sixth process has been opened with OP Open connection, an additional user is added to accommodate the six additional connections, and so on.

The maximum number of concurrent users is set by the 4D Server license which you have purchased.

Connecting to a 4D Server Database

4D Open allows you to connect to a 4D Server database located on a server. You have to select the database.

If you call OP Select 4D Server in your code, users will be presented with a dialog box that allows them to choose the 4D Server database. If you use OP Find 4D Server instead, you must pass the database name to the routine.

OP Select 4D Server displays the **Connect to data server** dialog box.
For a full discussion of opening connection with 4D Server, refer to the *4D Server Reference* manual.

OP Select 4D Server (netCompID; serverName; serverID; otherButton) → Longint

Parameter	Type		Description
netCompID	Longint	→	Network component reference number (always 2)
serverName	Text	←	Selected server name
serverID	Longint	←	Unique ID for selected server
otherButton	Boolean	→	To show or hide the "Other" button
Function result	Longint	←	Error code result for the function

Description

OP Select 4D Server allows you to graphically choose a 4D Server from the Connect to Data Server dialog box.

Pass 2 in netCompID. Starting from version 2003 of 4D Open for 4D, only the TCP/IP network component (reference number = 2) can be used.

The server name is returned in serverName. This name can be saved somewhere in the database to be later passed to OP Find 4D Server. serverName is equal to the IP address followed by a colon and the name of the database. For example, the following would be returned for a Customers DB database with the IP address 195.4.210.25:

"195.4.210.25:Customers DB".

The server reference number is returned in serverID. This reference number is then passed to OP Open connection to open a connection to the selected database.

Error Codes

If OP Select 4D Server executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
-10020	No server was selected while using OP Select 4D Server, (the Cancel button was clicked).
2	The Other button was clicked while using OP Select 4D Server.
10128	The 4D Open for 4th Dimension package has not been initialized.
10129	The network component was not found.
10131	The network component has not been initialized.
10154	This command cannot be executed right now.

Example

The following method allows you to choose a database from the list and open a connection to it.

```
C_STRING(255;$Server)
C_LONGINT($ErrCode;$ServerID)

⇒ $ErrCode:=OP Select 4D Server (2;$Server;$ServerID)
```

OP Find 4D Server (netCompID; serverName; serverID) → Longint

Parameter	Type		Description
netCompID	Longint	→	Network component reference number
serverName	String	→	Name of server to find
serverID	Longint	←	Unique ID for found server
Function result	Longint	←	Error code result for the function

Description

OP Find 4D Server searches for the database specified by netCompID and serverName.

To find a server using TCP/IP, serverName should be equal to the IP address of the server's computer. For example, you should pass the following for a database published on the IP address 192.9.200.13:

"192.9.200.13"

The server reference number is returned in serverID. You then pass this reference number to OP Open connection to open a connection to the selected database.

Error Codes

If the function executes successfully, OP Find 4D Server returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
-10021	No server was found while using OP Find 4D Server.
10128	The 4D Open for 4th Dimension plu-in has not been initialized.
10129	The network component was not found.
10131	The network component has not been initialized.
10154	This command cannot be executed right now.

Example

The following method finds a 4D Server database located at a specific IP address, and opens the connection.

```
C_STRING(80;$IPAddress)
C_LONGINT($Tcp;$ErrCode;$ServerID;vConnectID)

$IPAddress:="192.168.20.30"
$Tcp:=GetCompID ("TCP/IP")
```

```
⇒       $ErrCode:=OP Find 4D Server ($Tcp;$IPAddress;$ServerID)
        $ErrCode:=OP Open connection ($ServerID;vConnectID;"Arnaud's P4";
                                                    "Marcel Chombier";"";"DataSync")

           `... perform operations on the server

        $ErrCode:=OP Close connection (vConnectID)
        $ErrCode:=OP Delete 4D Server ($ServerID)
```

See Also

OP Delete 4D Server, OP Open connection, OP Select 4D Server.

OP Delete 4D Server (serverID) → Longint

Parameter	Type		Description
serverID	Longint	→	Unique ID of server to be erased
Function result	Longint	←	Error code result for the function

Description

OP Delete 4D Server clears the space in memory allocated to the process by the server.

This function clears the space in memory on the 4D Open client machine used to keep track of the network server location. Call this routine after you have closed all connections to a server.

If you apply this function to a server to which you are still connected, the routine does nothing and returns an error.

Error Codes

If the function was successful, OP Delete 4D Server returns 0. Otherwise this function returns one of the following errors:

Error Code	Description
10128	The 4D Open for 4th Dimension package has not been initialized.
10133	The server does not exist.
10134	The server is currently in use. It cannot be deleted.
10154	This command cannot be executed right now.

See Also

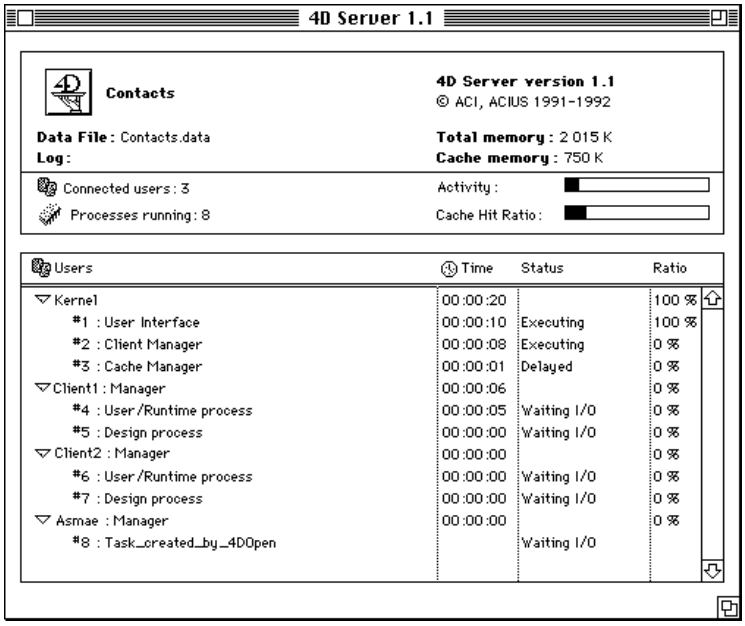
OP Find 4D Server, OP Select 4D Server.

OP Open connection (serverID; connectionID; station; userName; password; taskName) → Longint

Parameter	Type		Description
serverID	Longint	→	Unique ID of target server
connectionID	Longint	←	Unique ID for the established connection
station	String	→	Machine name for the server administration window
userName	String	→	4D user name
password	String	→	4D user password
taskName	String	→	Name of the process to be created on the server
Function result	Longint	←	Error code result for the function

Description

OP Open connection opens a connection to a server by passing the serverID, stationName, userName, password and taskName. This function returns the connection reference number in the connectionID parameter.



Error Codes

If the function executes successfully, OP Open connection returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-12	The maximum number of concurrent users for your 4D Server license has been reached.
-13	Your attempt to connect to 4D Server has not been successful due to a network problem, try again.
-108	Not enough memory to perform this operation.
-1277	Open connection request was denied (ADSP time-out error).
-9944*	The user does not belong to the 4D Open access group.
-9947*	The "Allow 4D Client connections only" check box has been checked in the Preferences dialog box of 4D Client.
-9956	4D Open version not compatible with 4D Server. Or, the "Allow 4D Client connections only" check box has been checked in the Preferences dialog box of 4D Client in version 1.1 of 4D Server.
-9978	Invalid password.
-9979	Unknown user.
10128	The 4D Open for 4th Dimension package has not been initialized.
10129	The network component was not found.
10133	The server does not exist.
10154	This command cannot be executed right now.

See Also

OP Close connection, OP Find 4D Server.

OP Close connection

OP Close connection (connectionID) → Number

Parameter	Type		Description
connectionID	Number	→	ID of connection to be closed
Function result	Number	←	Error code result for the function

Description

OP Close connection closes the connection to the server by passing connectionID.

Error Codes

If OP Close connection executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	This connection does not exist.
10154	This command cannot be executed right now.

See Also

OP Open connection.

5

Structure Information

The routines described in this chapter allow you to obtain information about the structure of a 4D Server database, such as the names, numbers and attributes of the tables and fields:

- OP Count tables - returns the number of tables in the 4D Server database.
- OP Get all tablenames - returns the names of all the tables in the 4D Server database.
- OP Get field properties - returns basic information about the table and the fields it contains.
- OP Get table properties - returns more extensive information about a table and the fields it contains.
- OP Get one field number - returns the table and field number for a specified field.
- OP Get all field numbers - returns the table and field numbers for the specified fields.
- OP Get one tablename - returns the name of the specified table.
- OP Count fields - returns the number of fields in a particular table.
- OP Cache structure - caches the 4D Server database structure in the client machine's memory.

OP Count tables (connectionID; numOfTables) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
numOfTables	Longint	←	Number of tables in server database
Function result	Longint	←	Error code result for the function

Description

OP Count tables returns in numOfTables the number of tables in the 4D Server database to which you have connected.

Error Codes

If the function OP Count tables executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

This example returns the number of files in the database. Then it uses this number to find the number of fields in each file.

```
C_LONGINT($i;vFileNum;vFieldNum)

`... connect to a server

⇒ $ErrCode:=OP Count tables($ConnectID;vFileNum)

ARRAY STRING(15;aFieldNum;vFileNum)

For($i;1;vFileNum)
   $ErrCode:=OP Count fields($ConnectID;$i;vFieldNum)
   aNumFields{$i}:=vFieldNum
End for
```

See Also

Count tables, OP Count fields.

OP Get all tablenames (connectionID; tableNames) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableNames	Array	←	Array of table names in the database
Function result	Longint	←	Error code result for the function

Description

OP Get all tablenames returns the names of all the tables in the 4D Server database in the tableNames array, which can be of type String or Text.

Error Codes

If the function OP Get all tablenames executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
10128	The 4D Open for 4th Dimension package has not been initialized.
10135	Invalid parameter type.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

This example fills the aFiles array with all the filenames in the database. The filenames are then displayed to the user in a dialog box.

```
     ARRAY STRING(15;aFiles;0)
⇒    $ErrCode:=OP Get all tablenames (vConnectID;aFiles)

     CenterWindow (150;191)
     DIALOG([Dialogs];"Show Files")
     CLOSE WINDOW

     If (aFiles>0)
        `… user selected a file
        `… perfom operation on selected file
     End if
```

See Also

OP Get one tablename, Table name.

OP Get field properties (connectionID; tableID; tableInvisible; fieldNames; fieldTypes; fieldLengths; fieldIndexed; fieldInvisible) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
tableInvisible	Longint	←	Table visibility
fieldNames	Array	←	Array of fields names for target table
fieldTypes	Array	←	Array of field types for target table
fieldLengths	Array	←	Array of field lengths for target table
fieldIndexed	Array	←	Array of field "index" properties for target table
fieldInvisible	Array	←	Array of field "visible" properties for target table
Function result	Longint	←	Error code result for the function

Description

OP Get field properties returns the name and type of all fields in the specified table. It also returns the length of all alpha fields in the table and indicates whether or not each field is indexed, and whether or not each field is invisible.

tableInvisible returns 1 if tableID is invisible. Otherwise, tableInvisible returns 0.

fieldNames must be an array of type String or Text.

fieldTypes must be an array of type String, Text, Real, Integer or Longint. The values returned in fieldTypes depend on the type of array passed:

Alpha Array	Numeric Array
"Alphanumeric"	0
"Real"	1
"Text"	2
"Picture"	3
"Date"	4
"Boolean"	6
"Subfile"	7
"Integer"	8
"Long Integer"	9
"Time"	11

fieldLengths can be an array of type String, Text, Real, Integer, or Long Integer. If the array is of type String or Text, the value passed is a numeric string. If the array is of type Real, Integer, or Long Integer, the value passed is a number. The values returned by Indexed vary depending on the type of array passed:

	Alpha Array	Numeric Array	Boolean Array
Field is indexed	"Indexed"	1	True
Field is not indexed	"" (empty string)	0	False

The values returned by InvisibleField vary depending on the type of array passed:

	Alpha Array	Numeric Array	Boolean Array
Field is invisible	"Invisible"	1	True
Field is not invisible	"" (empty string)	0	False

Error Codes

If OP Get fields properties executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
-9972	File number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10135	Invalid parameter type.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

In this example, arrays are used to store information about fields in a file.

```
C_LONGINT($File;$ErrCode)

ARRAY LONGINT(aTypes;0)      `…field types
ARRAY LONGINT(aLength;0)     `… field length (for alpha fields)
ARRAY LONGINT(aIndexes;0)    `…indexed?
ARRAY LONGINT(aInvisible;0)  `…invisible?
ARRAY STRING(15;aNames;0)    `…field names

$File:=1  `…get information on file 1

$ErrCode:=OP Load network component ($CompID)
$ErrCode:=OP Open connection ($ServerID;$ConnectID;"4D  3.2";"Designer";
                                                        $Password;"Nomad")
⇒ $ErrCode:=OP Get field properties ($ConnectID;$File;$Invisible;aNames;aTypes;
                                        aLength;aIndexes; aInvisible)
```

See Also

GET FIELD PROPERTIES.

OP Get table properties (connectionID; tableID; tableInvisible; tableLeft; tableTop; tableRight; tableBottom; fieldNames; fieldTypes; fieldLength; fieldIndexed; fieldUnique; fieldInvisible; fieldEnterable; fieldModifiable; fieldMandatory; fieldRelatedTable; fieldRelatedField; fieldRelatedWild; fieldManyToOne; fieldOneToMany; fieldDelControl; fieldWhatControl; fieldAutoAssign) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
tableInvisible	Longint	←	Table "visible" property
tableLeft	Longint	←	Table left graphical coordinate
tableTop	Longint	←	Table top graphical coordinate
tableRight	Longint	←	Table right graphical coordinate
tableBottom	Longint	←	Table bottom graphical coordinate
fieldNames	Array	←	Array fo field names for target table
fieldTypes	Array	←	Array of field types for target table
fieldLength	Array	←	Array of field lengthes for target table
fieldIndexed	Array	←	Array of "indexed" properties for target tables
fieldUnique	Array	←	Array of "unique" properties for target table
fieldInvisible	Array	←	Array of "visible" properties for target table
fieldEnterable	Array	←	Array of "Enterable" properties for target table
fieldModifiable	Array	←	Array of "Modifiable" properties for target table
fieldMandatory	Array	←	Array of "Mandatory" properties for target table
fieldRelatedTable	Array	←	Array of related table numbers for target table
fieldRelatedField	Array	←	Array of related field numbers for target table
fieldRelatedWild	Array	←	Array of "field used for wildcard" numbers for target table
fieldManyToOne	Array	←	Array of "many to one" automatic relation property for target table
fieldOneToMany	Array	←	Array of "one to many" automatic relation property for target table
fieldDelControl	Array	←	Array of "deletion control" property for target table
fieldWhatControl	Array	←	Array of "deletion control action" property for target table
fieldAutoAssign	Array	←	Array of "auto-assign related value" property for target table
Function result	Longint	←	Error code result for the function

Description

OP Get table properties returns extensive information about a table and the fields it contains.

fieldTypes can be a Text, String, Integer, Long Integer, or Real array. The values returned in fieldTypes depend on the type of array passed:

Alpha Array	Numeric Array
"Alphanumeric"	0
"Real"	1
"Text"	2
"Picture"	3
"Date"	4
"Boolean"	6
"Subfile"	7
"Integer"	8
"Long Integer"	9
"Time"	11

fieldLength can be Text, String, Integer, Long Integer, or Real arrays. If the array is of type Text or String, the lengths are returned in text form.

fieldRelatedTable, fieldRelatedField, and fieldRelatedWild can be Integer, Long Integer, or Real arrays. fieldRelatedTable only returns a related table number if the file described by tableID is the One table in a relation to another table.

fieldIndexed, fieldInvisible, fieldUnique, fieldEnterable, fieldModifiable, and fieldMandatory can be String, Text, Integer, Long Integer, Real, or Boolean arrays. The values returned depend on the types of arrays passed, based on the following table:

	Alpha Array	Numeric Array	Boolean Array
Field is indexed	"Indexed"	1	True
Field is not indexed	"" (empty string)	0	False
Field is invisible	"Invisible"	1	True
Field is not invisible	"" (empty string)	0	False
Field is unique	"Unique"	1	True
Field is not unique	"" (empty string)	0	False
Field is enterable	"Enterable"	1	True
Field is not enterable	"" (empty string)	0	False
Field is modifiable	"Modifiable"	1	True
Field is not modifiable	"" (empty string)	0	False
Field is mandatory	"Mandatory"	1	True
Field is not mandatory	"" (empty string)	0	False

fieldManyToOne, fieldOneToMany, fieldDelControl, fieldWhatControl, and fieldAutoAssign can be Integer, Long Integer, Real, or Boolean arrays. 4D Open will return 1 (**true**) or 0 (**false**) in numeric arrays and "True" or "False" in Boolean arrays.

Error Codes

If OP Get table properties executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
-9972	File number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10135	Invalid parameter type.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

The following code lists the invisible files and fields in a database:

```
C_LONGINT($Invisible;$L;$T;$R;$B;$ErrCode;$FileID;$FldID)

ARRAY STRING(31;artName;0)
   ` ConnID is a valid connection reference number
$ErrCode:=OP Get all tablenames (ConnID;artName)

If ($ErrCode=0)
   ARRAY STRING(31;arFldName;0)  `Can be String or Text
   ARRAY INTEGER(arFldType;0)  ` Can be Integer, LongInt, Real, String or Text
   ARRAY INTEGER(arFldLen;0)  ` Can be Integer, LongInt, Real, String or Text
   ARRAY INTEGER(arFldInd;0) ` Can be Integer, LongInt, Real, Boolean, String or Text
   ARRAY BOOLEAN(arFldInv;0) `Can be Integer, LongInt, Real, Boolean, String, Text
   ARRAY INTEGER(arFldUni;0) ` Can be Integer, LongInt, Real, Boolean, String, Text
   ARRAY BOOLEAN(arFldEnt;0) `Can be Integer, LongInt, Real, Boolean, String, Text
   ARRAY BOOLEAN(arFldMod;0) `Can be Integer, LongInt, Real, Boolean, String, Text
   ARRAY BOOLEAN(arFldMan;0) `Can be Integer, LongInt, Real, Boolean, String, Text
   ARRAY INTEGER(arFldRelFl;0)  ` Can be Integer, LongInt or Real
   ARRAY INTEGER(arFldRelFd;0)  ` Can be Integer, LongInt or Real
   ARRAY INTEGER(arFldDis;0)  ` Can be Integer, LongInt or Real
   ARRAY BOOLEAN(arFldAutM2O;0  ` Can be Integer, LongInt, Real or Boolean
   ARRAY BOOLEAN(arFldAutO2M;0)  ` Can be Integer, LongInt, Real or Boolean
   ARRAY BOOLEAN(arFldDel;0)  ` Can be Integer, LongInt, Real or Boolean
   ARRAY BOOLEAN(arFldRej;0)  ` Can be Integer, LongInt, Real or Boolean
   ARRAY BOOLEAN(arFldUpd;0)  ` Can be Integer, LongInt, Real or Boolean
```

```
      For ($tableID;1;Size of array(artName))
⇒        $ErrCode:=OP Get table properties (ConnID;$tableID;$Invisible;$L;$T;$R;$B;
                        arFldName;arFldType;arFldLen;arFldInv;arFldInd;arFldUni;arFldEnt;
                          arFldMod;arFldMan;arFldRelFl;arFldRelFd;arFldDis;arFldAutM2O;
                                                  arFldAutO2M;arFldDel;arFldRej;arFldUpd)
         If ($ErrCode=0)
            If ($Invisible#0)
               MESSAGE("The table ["+artName{$tableID}+"] is invisible.")
            Else
               For ($FldID;1;Size of array(arFldName))
                  If (arFldInv{$FldID})
                     MESSAGE("The field ["+artName{$tableID}+"]"+arFldName{$FldID}+
                                                                          " is invisible.")
                  End if
               End for
            End if
         End if
      End for
   End if
```

See Also

OP Get all tablenames.

OP Get one field number (connectionID; tableField; tableID; fieldID) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableField	String	→	Name of the field in "[table]field" format
tableID	Longint	←	Number of the table in the database
fieldID	Longint	←	Number of the field in the table
Function result	Longint	←	Error code result for the function

Description

OP Get one field number returns tableID and fieldID for [TableName]fieldName passed in tableField. The table and field name are not case-sensitive.

Error Codes

If OP Get one field number executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
-9972	File number is out of range.
-9981	Invalid table/field number definition array sent.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10146	Invalid field name(s).
10154	This command cannot be executed right now.

Example

In this example, the file and field number are returned for the “[Customer]Full name” field:

```
C_LONGINT($ErrCode;vTable;vField)

⇒ $ErrCode:=OP Get one field number (vConnectID;"[Customer]Full Name";vTable;vField)
If ($ErrCode=0)
   ALERT("Table number: "+String(vTable)+Char(13)+"Field number: "+String(vField))
End if
```

See Also

Field, OP Get all field numbers, OP Get field properties.

OP Get all field numbers (connectionID; tableFields; tableIDs; fieldIDs) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with the server
tableFields	Array	→	Array of field names in "[table]field" format
tableIDs	Array	←	Array of table numbers in the database
fieldIDs	Array	←	Array of field IDs in their tables
Function result	Longint	←	Error code result for the function

Description

OP Get all field numbers returns in tableIDs and in fieldIDs the table and field numbers for the names of the tables and fields you pass in tableFields.

tableFields must be an array of type Text or String. You must use the “[Table]Field” name format.

The table and field names are not case-sensitive.

tableIDs and fieldIDs must be arrays of type Integer or Longint or Real.

Error Codes

If OP Get all field numbers executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
-9981	Invalid table/field number definition array sent.
10128	The 4D Open for 4th Dimension package has not been initialized.
10135	Invalid parameter type.
10136	The connection does not exist.
10146	Invalid field name(s).
10154	This command cannot be executed right now.
10159	The array(s) contain or will contain too many elements.

Example

In this example, the file and field numbers are returned for the fields passed in the aFldNames array:

```
C_LONGINT($ErrCode)

ARRAY LONGINT(aTables;6)
ARRAY LONGINT(aFields;6)
ARRAY STRING(15;aFldNames;6)

aFldNames{1}:="[People]First Name"
aFldNames{2}:="[People]Last Name"
aFldNames{3}:="[Company]Address"
aFldNames{4}:="[Company]City"
aFldNames{5}:="[Company]ZIP"
aFldNames{5}:="[Invoices]Amount"

⇒ $ErrCode:=OP Get all field numbers (vConnectID;aFldNames;aTables;aFields)
```

See Also

Field, OP Count fields, OP Get field properties, OP Get one field number.

OP Get one tablename (connectionID; tableID; nameOfTable) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of table in the database
nameOfTable	String	←	Name of the taget table
Function result	Longint	←	Error code result for the function

Description

OP Get one tablename returns in nameOfTable the name of tableID.

Error Codes

If OP Get one filename executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

See Also

OP Get all tablenames, Table name.

OP Count fields (connectionID; tableID; nbOfFields) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
nbOfFields	Longint	←	Number of fields for target table
Function result	Longint	←	Error code result for the function

Description

OP Count fields returns in nbOfFields the number of fields in tableID.

Error Codes

If OP Count fields executes successfully, it return 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

See Also

Count fields, OP Count tables.

OP Cache structure (connectionID; cacheStructure) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
cacheStructure	Longint	→	Cache structure option (1=Yes)
Function result	Longint	←	Error code result for the function

Description

If cacheStructure is not zero, OP Cache structure instructs 4D Open to start caching the 4D Server database structure for this connection (AND for all the other connections to this database) in the client machine's memory. In this case, ALL the routines that need structure information will use the cached information.

To stop caching the structure for a particular database, call OP Cache structure with cacheStructure equal to zero. Caching is off by default.

Caching is very useful for reducing requests sent over the network. However, this means that you expect to have a "frozen" database structure. Unlike the .rex file scheme, a change to the actual structure won't be reflected in the cached information. In addition, caching is local to a session; no structure information is kept on disk.

Before calling OP Cache structure, you must call OP Set option (3;1) to set "cache structure information" to on.

Error Codes

If OP Cache structure executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

See Also

OP Get option, OP Set option.

6

Query and Order

Query and Order, Introduction

When managing data, you can select a group of records with which you want to work. To do so, you can either select all the records in the table or perform a query that selects specific records.

There is always one selection of records per table per process. The current selection is the set of records most recently selected. To save the current selection, you can use named selections as discussed in the Named Selections section.

You can perform single and multiple queries. Single queries contain one search criterion while multiple queries contain two or more query criteria.

Multiple Queries

Querying on two or more fields is called a compound or multiple query. When you perform a compound query, you combine separate query conditions using a logical operator. The logical operator tells 4th Dimension how to combine the results of the individual queries.

The logical operators let you create compound queries conditions such as “Find the employees located in New York or in California.” The query conditions for this search would be as follows: When this query is executed, 4D Server finds all the employees located in either New York or California.

When querying a table, you can query the entire table or only the current selection. You can also choose between a simple query that uses a single query criterion or a more complex query that uses multiple query criteria.

The Query and Order routines are:

- OP Single query - performs a simple query on the entire file.
- OP Single query selection - performs a simple query on the current selection.
- OP Single order by - orders the records in the selection on one field.
- OP Multi query - performs a more complex query on the entire file.
- OP Multi query selection - performs a more complex query on the current selection.
- OP Multi order by - orders the records in the selection on more than one field.

OP Single query (connectionID; tableID; fieldID; queryOperator; queryValue; recordsFound) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
fieldID	Longint	→	Number of the field in the table
queryOperator	String	→	Query operator
queryValue	Pointer	→	Pointer to object containing query value
recordsFound	Longint	←	Number of records in the resulting selection
Function result	Longint	←	Error code result for the function

Description

OP Single query queries tableID for queryValues in fieldID. The query is performed on all the records in tableID. After the query, recordsFound indicates the number of records in the new current selection.

qureyValue is a pointer to a variable or field that contains the value with which to perform the query. This value can begin with, contain or end with the "@" symbol for wildcard queries. The pointer passed for queryValue can be a pointer to a field or a process or interprocess variable. If necessary, queryValue is converted.

To query on a Boolean field, you must pass "**false**" for False and "**true**" for True.

The query operators are as follows:

Operator	Description
"="	Equal
"#"	Not equal
">"	Greater than
">="	Greater than or equal to
"<"	Less than
"<="	Less than or equal to

Error Codes

If OP Single query executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9969	Invalid field type requested.
-9971	Field number is out of range.
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10144	The 4D pointer is equal to NIL.
10145	A 4D pointer was expected.
10151	Invalid search operator.
10153	Unable to convert this type of data.
10154	This command cannot be executed right now.

Example

This example searches for records in the [Invoices] table whose grand total is greater than 10,000.

```
C_LONGINT(vRecords; $ErrCode)
C_REAL(vValue)
C_LONGINT(vTable;vField)

vValue:=10000
$errCode:=OP Get one field number (vConnectID;"[Invoices]Total";vTable;vField)
⇒ $ErrCode:=OP Single query (vConnectID;vTable;vField;">";»vValue;vRecords)

If ($ErrCode=0)
   ALERT(String(vRecords)+" record(s) were found.")
End if
```

See Also

OP Get one field number, OP Multi query, OP Single order by, ORDER BY.

OP Single query selection (connectionID; tableID; fieldID; queryOperator; queryValue; recordsFound) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
fieldID	Longint	→	Number of the field in the table
queryOperator	String	→	Query operator
queryValue	Pointer	→	Pointer to the object containing the query value
recordsFound	Longint	←	Number of records in the resulting selection
Function result	Longint	←	Error code result for the function

Description

OP Single query selection queries tableID for queryValues in fieldID. The query is performed on the records in tableID's current selection. After the query, recordsFound indicates the number of records in the new current selection.

queryValue is a pointer to a variable or field that contains the value with which to perform the query. This value can begin with, contain or end with the "@" symbol for wildcard queries. The pointer passed for queryValue can be a pointer to a field, or a process or interprocess variable. If necessary, queryValue is converted.

To query on a Boolean field, you must pass "**false**" for False and "**true**" for True.

The query operators are as follows:

Operator	Description
"="	Equal
"#"	Not equal
">"	Greater than
">="	Greater than or equal to
"<"	Less than
"<="	Less than or equal to

Error Codes

If OP Single query selection executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9969	Invalid field type requested.
-9971	Field number is out of range.
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10144	The 4D pointer is equal to NIL.
10145	A 4D pointer was expected.
10151	Invalid search operator.
10153	Unable to convert this type of data.
10154	This command cannot be executed right now.

Example

This example searches for records in the [Invoices] file whose grand total is greater than 10,000. It then queries the selection for invoices having dates greater than January 1 of the current year.

```
C_LONGINT(vRecords; $ErrCode)
C_REAL(vValue)
C_LONGINT(vTable;vField1;vField2)

$errCode:=OP Get one field number (vConnectID;"[Invoices]Total";vTable;vField1)
$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice date";vTable;
                                                                    vField2)
vValue:="10000"
$errCode:=OP Single query (vConnectID;vTable;vField1;">";»vValue;vRecords)
vValue:="01/01/" + String ( year of ( Current date );"####")
⇒ $errCode:=OP Single query selection (vConnectID;vTable;vField2;">";»vValue;vRecords)

If ($ErrCode=0)
   ALERT(String(vRecords)+" record(s) were found.")
End if
```

See Also

OP Get one field number, OP Multi query, OP Single order by, OP Single query, QUERY SELECTION.

OP Single order by

Query and Order

version 1.5

OP Single order by (connectionID; tableID; fieldID; orderDirection) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
fieldID	Longint	→	Number of the field in the table
orderDirection	String	→	Order by direction (">" is ascending)
Function result	Longint	←	Error code result for the function

Description

OP Single order by orders the tableID table's current selection by the values in the field specified by fieldID.

You specify the sorting order by passing either “>” (for ascending order) or “<” (for descending order) to orderDirection.
If the field on which you are ordering is indexed, 4D Server uses that index.

Error Codes

If OP Single order by executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9969	Invalid field type requested.
-9971	Field number is out of range.
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.
10155	Invalid sorting order.

Example

This example orders the current selection of invoices by date in ascending order:

```
      C_LONGINT (vTable;vField)
      C_LONGINT($errCode)
      $errCode:=OP Get one field number (vDonnectID;"[Invoices]Invoice date";vTable;vField)
⇒     $errCode:=OP Single order by (ConnectID;vTable;vField;">")
```

See Also

OP Get one field number, OP Multi order by, OP Multi query, OP Single query, ORDER BY.

OP Multi query

Query and Order

version 1.5

OP Multi query (connectionID; tableID; tableIDs; fieldIDs; logicalOperators; queryOperators; queryValues; recordsFound) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
tableIDs	Array	→	Array of related tables numbers in the database
fieldIDs	Array	→	Array of related field numbers in their tables
logicalOperators	Array	→	Query lines logical conjunctors
queryOperators	Array	→	Array of query operators
queryValues	Array	→	Array of query values
recordsFound	Longint	←	Numbers of records in resulting selection
Function result	Longint	←	Error code result for the function

Description

OP Multi query queries tableID using fields (from multiple tables) passed in fieldIDs. The new current selection is built from all the records in tableID. The maximum number of elements for any array is 20. If you pass an array with more than 20 elements, only the first 20 elements will be used, but no error will be returned.

tableIDs and fieldIDs are arrays of type Integer or Longint, containing the table and field numbers of the tables and fields to query.

logicalOperator and queryOperator are arrays of type Integer, Longint, Real, String or Text, containing the logical and query operators.

The logical operators are as follows:

Description	Alpha Array	Numeric Array
And	"&"	1
Or	"|"	2
Except	"#"	0

The logical operator connects each query line to the result obtained by the previous line(s). Therefore, the logical operator for the first search line is simply ignored.

The query operators are as follows:

Description	Alpha Array	Numeric Array
Equal	“=”	1
Not equal	“#”	2
Greater than	“>”	3
Greater than or equal to	“>=”	4
Less than	“<”	5
Less than or equal to	“<=”	6

queryValue is an array of type String or Text that contains one or more values that can begin with, contain or end with the “@” symbol for wildcard queries.

Values for each data type are passed as strings. For example, to query for a date, pass “12/03/97” for December 3, 1997. To query for a Boolean value, pass “**false**” for False and “**true**” for True.

Error Codes

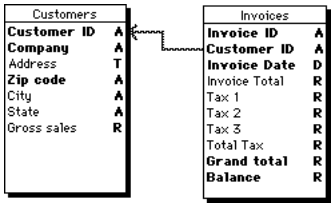
If OP Multi query executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9969	Invalid field type requested.
-9971	Field number is out of range.
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10135	Invalid parameter type.
10136	The connection does not exist.
10151	Invalid query operator.
10152	Invalid logical operator.
10153	Unable to convert this type of data.
10154	This command cannot be executed right now.
10156	Empty search or sort definition.

Note: If there is an error due to an element of one of the arrays that you have passed, OP Multi query sets the selected element number of each of the arrays to the number of the faulty element. For example, if there is an error in the fifth element of one of the arrays, the selected element number of all of the arrays is set to 5.

Example

This example searches for invoices concerning customers located in Ohio or Texas with an invoice total greater than 2000. It uses the following structure:




```
C_LONGINT($ErrCode;$TrgTable;vRecords)
ARRAY LONGINT(arTableID;3)
ARRAY LONGINT(arFieldID;3)
ARRAY TEXT(arLogicalOp;3)
ARRAY TEXT(arQueryOp;3)
ARRAY TEXT(arValues;3)

$TrgTable:=2          ` We want to build a selection of records on [Invoices]
arTableID{1}:=1       `      [Customers]
arTableID{2}:=1       `      [Customers]
arTableID{3}:=2       `     [Invoices]

arFieldID{1}:=6       `      [Customers]State
arFieldID{2}:=6       `      [Customers]State
arFieldID{3}:=4       `      [Invoices]Invoice total

arQueryOp{1}:="="     `      [Customers]State =
arQueryOp{2}:="="     `      [Customers]State =
arQueryOp{2}:=">"     `      [Invoices]Invoice total >

arValues{1}:="Texas"  `      [Customers]State = "Texas"
arValues{2}:="Ohio"   `      [Customers]State = "Ohio"
arValues{3}:="2000"   `      [Invoices]Invoice total > 2000

arLogicalOp{1}:=""    `      [Customers]State = "Texas"
arLogicalOp{2}:="|"   ` or   [Customers]State = "Ohio"
arLogicalOp{2}:="&"   ` and  [Invoices]Invoice total > 2000

⇒ $ErrCode:=OP Multi query (vConnectID;$TrgFile;arFileID;arFieldID;
     arLogicalOp;arQueryOp;arValues;vRecords)
  If ($ErrCode=0)
     ALERT(String(vRecords)+" record(s) were found.")
  End if
```

See Also

OP Multi query selection, OP Single order by, OP Single query, QUERY.

OP Multi query selection (connectionID; tableID; tableIDs; fieldIDs; logicalOperators; queryOperators; queryValues; recordsFound) → Number

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
tableIDs	Array	→	Array of related tables numbers in the database
fieldIDs	Array	→	Array of related field numbers in their tables
logicalOperators	Array	→	Query lines logical conjunctors
queryOperators	Array	→	Array of query operators
queryValues	Array	→	Array of query values
recordsFound	Longint	←	Numbers of records in resulting selection
Function result	Number	←	Error code result for the function

Description

OP Multi query selection queries tableID using fields (from multiple tables) passed in fieldIDs. The new current selection is build from tableID 's current selection.

The maximum number of elements for any array is 20. If you pass an array with more than 20 elements, only the first 20 elements of the array will be used, but no error will be returned.

tableIDs and fieldIDs are arrays of type Integer or Longint, containing the table and field numbers of the tables and fields to query.

logicalOperator and queryOperator are arrays of type Integer, Longint, Real, String or Text, containing the logical and query operators.

The logical operators are as follows:

Description	Alpha Array	Numeric Array
And	“&”	1
Or	“|”	2
Except	“#”	0

The logical operator connects each query line to the result obtained by the previous line(s). Therefore, the logical operator for the first search line is simply ignored.

The query operators are as follows:

Description	Alpha Array	Numeric Array
Equal	"="	1
Not equal	"#"	2
Greater than	">"	3
Greater than or equal to	">="	4
Less than	"<"	5
Less than or equal to	"<="	6

queryValue is an array of type String or Text that contains one or more values that can begin with, contain or end with the "@" symbol for wildcard queries.

Values for each data type are passed as strings. For example, to query for a date, pass "12/03/97" for December 3, 1997. To query for a Boolean value, you must pass "**false**" for False and "**true**" for True.

Error Codes

If OP Multi query executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9969	Invalid field type requested.
-9971	Field number is out of range.
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10135	Invalid parameter type.
10136	The connection does not exist.
10151	Invalid query operator.
10152	Invalid logical operator.
10153	Unable to convert this type of data.
10154	This command cannot be executed right now.
10156	Empty search or sort definition.

Note: If there is an error due to an element of one of the arrays that you have passed, OP Multi query sets the selected element number of each array to the number of the faulty element. For example, if there is an error in the fifth element of one of the arrays, the selected element number of all of the arrays is set to 5.

Example

This example searches for invoices concerning customers located in Ohio or Texas on a selection of invoices prviously made by the user.

```
C_LONGINT($ErrCode;$TrgTable;vRecords)
ARRAY LONGINT(arTableID;0)
ARRAY LONGINT(arFieldID;0)
ARRAY TEXT(arLogicalOp;0)
ARRAY TEXT(arQueryOp;0)
ARRAY TEXT(arValues;0)

UserSelectInvoices  `Display a dialog that let the user specify criteria

   ` Perform the query specified by the user
OP Multi query(vConnectID;2;arFileID;arFieldID;arLogicalOp;arQueryOp;
                                                        arValues;vRecords)

ARRAY LONGINT(arTableID;3)
ARRAY LONGINT(arFieldID;3)
ARRAY TEXT(arLogicalOp;2)
ARRAY TEXT(arQueryOp;2)
ARRAY TEXT(arValues;2)

$TrgTable:=2          ` We want to build a selection of records on [Invoices]

arTableID{1}:=1       `      [Customers]
arTableID{2}:=1       `      [Customers]
arFieldID{1}:=6       `      [Customers]State
arFieldID{2}:=6       `      [Customers]State
arQueryOp{1}:="="     `      [Customers]State =
arQueryOp{2}:="="     `      [Customers]State =
arValues{1}:="Texas"  `      [Customers]State = "Texas"
arValues{2}:="Ohio"   `      [Customers]State = "Ohio"
arLogicalOp{1}:=""    `      [Customers]State = "Texas"
arLogicalOp{2}:="|"   ` or   [Customers]State = "Ohio"

` Then queries the user-defined selection with new criteria

⇒  $ErrCode:=OP Multi query selection (vConnectID;$TrgFile;arFileID;arFieldID;
      arLogicalOp;arQueryOp;arValues;vRecords)

If ($ErrCode=0)
   ALERT(String(vRecords)+" record(s) were found.")
End if
```

See Also

OP Multi query, OP Single order by, OP Single query, QUERY SELECTION.

OP Multi order by

Query and Order

version 1.5

OP Multi order by (connectionID; tableID; tableIDs; fieldIDs; orderDirections) → Longint

Parameter	Type		Description
connectionID	Longint		
tableID	Longint	→	Number of table to order
tableIDs	Array	→	Array of table IDs in the database
fieldIDs	Array	→	Array of field IDs in their tables
orderDirections	Array	→	Array of ordering directions (">" is ascending)
Function result	Longint	←	Error code result for the function

Description

OP Multi order by orders the target table targetFileID 's current selection by the values in the fields specified by the tableIeID and fieldID arrays.

The maximum number of elements for any array is 20. If you pass an array with more than 20 elements, only the first 20 elements of the array will be used, but no error will be returned.

tableIDs and fieldIDs are arrays of type Integer, Longint or Real.

orderDirections is an array of type String, Text, Real, Integer, Long Integer, or Boolean. If orderDirections is of type Text or String, the array should contain one of two values for each table/field to order: “>” to sort in ascending order or “<” to sort in descending order. If orderDirections is a numeric array, a positive value indicates an ascending sort, a zero or negative value indicates a descending sort. If orderDirections is a Boolean array, "**true**" indicates an ascending sort, while "**false**" indicates a descending sort.

If parameter checking is on and orderDirections is a Text or String array, any value other than "<" or ">" generates an error. If parameter checking is off, any value not starting with "<" is assumed to indicate ascending order.

4th Dimension performs a sequential sort on the fields chosen even if they are all indexed.

Error Codes

If OP Multi order by executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
-9971	Field number is out of range.
-9972	Tale number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10135	Invalid parameter type.
10136	The connection does not exist.
10154	This command cannot be executed right now.
10155	Invalid sorting order.
10156	Empty search or sort definition.

Note: If there is an error due to an element of one of the arrays that you have passed, OP Multi order by sets the selected element number of each of the arrays to the number of the faulty element. For example, if there is an error in the fifth element of one of the arrays, the selected element number of all of the arrays is set to 5.

Example

This example orders the selection of invoices by customer company name and invoice date. This example uses the file structure shown in the example for OP Multi query.

```
C_LONGINT($ErrCode;$TrgFile)

ARRAY LONGINT(aTableID;2)
ARRAY LONGINT(aFieldID;2)

ARRAY STRING(1;aOrder;2)

$Trgtable:=2   `...we want to sort the invoice table

aTableID{1}:=1     ` [Customers]
aTableID{2}:=1     ` [Invoices]

aFieldID{2}:=2     ` [Customers]Company
aFieldID{3}:=3     ` [Invoices]Date

aOrder{1}:=">"
aOrder{2}:=">"

⇒   $ErrCode:=OP Multi order by (vConnectID;$TrgFile;aFileID;aFieldID;aOrder)
```

See Also

OP Get one field number, OP Multi query, OP Order by, OP Single query, ORDER BY.

7

Selections

The routines described in this chapter allow you to manipulate selections of records as well place them in arrays:

- OP Records in table - returns the number of records in the specified table.
- OP Records in selection - returns the number of records in the current selection of the specified table.
- OP All records - selects all the records in the specified table.
- OP Reduce selection - creates a selection of arrays by reducing the current selection to the number you pass.
- OP Delete selection - deletes the current selection of records.
- OP Many to one join - creates a selection of records in the Many table.
- OP One to many join - creates a selection of records in the One table.
- OP Scan index - retrieves a selection of records from the table by specifying a number of records from the beginning or the end of the index.

OP Records in table (connectionID; tableID; recordsInTable) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
recordsInTable	Longint	←	Number of records in the table
Function result	Longint	←	Error code result for the function

Description

OP Records in table returns the number of records in table tableID in the recordsInTable variable.

Error Codes

If OP Records in table executes successfully, it return 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

```
      C_LONGINT (vTableID;vFieldID)
      C_LONGINT (vRecords)
      C_LONGINT ($errCode;$nbHours)
         ` Get table ID
      $errCode:=OP Get one field number(vConnectID;"[Invoices]Amount";vTableID;vFieldID)
         ` Get number of records in the table
⇒     $errCode:=OP Records in table (vConnectID;vTableID;vRecords)
         ` Compute average time to process entire file
      $nbHours:=vRecords*<>avgTimePerRecord
      If ($nbHours > 8) ` If more than eight hours to process, get user's OK
         CONFIRM ("This operation will take more than eight hours"+Char (13)+"Proceed ?")
         If (OK=1)
            processInvoices
         End if
      End if
```

See Also

OP Get one field number, OP Records in selection, Records in table.

OP Records in selection

Selections

version 1.5

OP Records in selection (connectionID; tableID; recordsInSelection) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Table ID in the database
recordsInSelection	Longint	←	Number of records in target table current selection
Function result	Longint	←	Error code result for the function

Description

OP Records in selection returns the number of records in the current selection of tableID in the recordsInSelection variable.

Error Codes

If OP Records in selection executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

```
C_LONGINT (vTableID;vFieldID)
C_LONGINT (vRecords)
C_LONGINT ($errCode;$nbHours)

UserSelectInvoices  `Display a dialog that lets the user specify criteria
UserQueryRun        ` Perform the query specified by the user

   ` Get number of records in the table
⇒  $errCode:=OP Records in selection (vConnectID;2;vRecords)

   ` Compute average time to process user selection
$nbHours:=vRecords*<>avgTimePerRecord
```

```
If ($nbHours > 8) ` If more than eight hours to process, get user's OK
   CONFIRM ("This operation will take more than eight hours"+Char (13)+"Proceed ?")
   If (OK=1)
      processInvoices
   End if
End if
```

See Also

OP All records, OP Multi query, OP Records in table, OP Single query, OP Use named selection, Records in selection.

OP All records (connectionID; tableID) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Table ID in the database
Function result	Longint	←	Error code result for the function

Description

OP All records selects all the records in table tableID. The new current selection contains all the records of tableID that were there when the command was executed.

Error Codes

If OP All records executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

```
C_LONGINT ($errCode;$nbHours)

UserSelectInvoices   `Display a dialog that let the user specify criteria

Case of
  :( userClickedQuery= 1) ` User clicked on 'query' button
    UserQueryRun ` Perform the query specified by the user
  :( userClickedCancel=1) `User cancels query request
    $errCode:= OP Reduce selection (vConnectID;vTableID;0)
  :( userClickedAllInvoices=1) `User chose to process all invoices
⇒   $errCode:=OP All records (vConnectID;vTableID)
End case
```

```
  ` Get number of records in the table
$errCode:=OP Records in selection (vConnectID;vTableID;vRecords)

  ` Compute average time to process user selection
$nbHours:=vRecords*<>avgTimePerRecord

If ($nbHours > 8) ` If more than eight hours to process, get user's OK
  CONFIRM ("This operation will take more than eight hours"+Char (13)+ "Proceed ?")
  If (OK=1)
    processInvoices
  End if
End if
```

See Also

ALL RECORDS, OP Records in selection, OP Records in table, OP Reduce selection, OP Single query.

OP Reduce selection

Selections

version 1.5

OP Reduce selection (connectionID; tableID; selectionSize) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
selectionSize	Longint	→	New size for the target table's current selection
Function result	Longint	←	Error code result for the function

Description

OP Reduce selection creates a new selection of records for table tableID. The new selection contains the first selectionSize records from previous current selection in table tableID.

If selectionSize is greater than the number of records in the current selection, all the records in the selection are selected.

Error Codes

If OP Reduce selection executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

This example finds the average amount of the top ten invoices of last year.

```
C_LONGINT(vRecords; $ErrCode)
C_REAL(vValue;vMeanAmount)
C_LONGINT(vTable;vFieldDate;vFieldAmount)

$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice date";vTable;
                                                                vFieldDate)
$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice amount";vTable;
                                                              vFieldAmount)
```

```
   `Compute last year's 1st of January
vValue:="01/01/" + String ( year of ( Current date ) -1;"####")

   ` Query for last year's invoices
$errCode:=OP Single query (vConnectID;vTable;vFieldDate;">";->vValue;vRecords)

   `Order by decreasing amount (largest amount is first of selection)
$errCode:=OP Single order by (vConnectID;vTable;vFieldAmount;"<")

   ` Focus on the 10 largest invoices
⇒  $errCode:=OP Reduce selection (vConnectID;vTable;10)

   `compute mean value
$errCode:=OP Sum (vConnectID;vTable;vFieldAmount;vMeanAmount)

ALERT (" Mean value for last year's top ten invoices is :"+ String (vMeanAmount;
                                                    "### ### ###.##"))
```

See Also

OP All records, OP Single order by, OP Records in selection, OP Single query, REDUCE SELECTION.

OP Delete selection (connectionID; tableID) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
Function result	Longint	←	Error code result for the function

Description

OP Delete selection deletes all the records of the current selection of tableID. If there are no records in the current selection, this function has no effect.

Error Codes

If OP Delete selection executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

This example selects, archives and deletes all invoices from three years ago.

```
C_LONGINT(vRecords; $ErrCode)
C_STRING (50;vValue)
C_LONGINT(vTable;vFieldDate)

$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice date";vTable;
                                                                    vFieldDate)
      ` Compute 31st of December of three years ago
vValue:="12/31/" + String ( Year of ( Current date ) -3 ;"####")
   ` Query for that year's invoices
$errCode:=OP Single query (vConnectID;vTable;vFieldDate;"<=";->vValue;vRecords)
ArchiveInvoices
   `Now we can delete the old invices
⇒  $errCode:=OP Delete selection (vConnectId;vTable)
```

See Also

DELETE SELECTION, OP All records, OP Records in selection, OP Single query.

OP Many to one join

Selections

version 1.5

OP Many to one join (connectionID; manyTableID; oneTableID) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
manyTableID	Longint	→	Number of the Many file in the database
oneTableID	Longint	→	Number of the One file in the database
Function result	Longint	←	Error code result for the function

Description

OP Many to one join creates a new selection of records in oneTableID based on the selection of records in manyTableID. For example, if you have a selection of records in the many table, you can retrieve the records in the One table that are related to that selection.

This command can work across several levels of relations. For example, according to the [Customers]/ [Invoices] table structure, a call to OP Many to one join with the Many file [Invoices] and the One file [Customers] selects the customers related to the selected records in [Invoices].

OP Many to one join works with both manual and automatic relations.

Error Codes

If OP Many to one join executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

This example prints a list of the customers who received the 10 largest invoices for the current year:

```
C_LONGINT(vRecords; $ErrCode)
C_STRING (10;vValue)
C_LONGINT(vTableInvoices;vTableCustomers;vFieldDate;vFieldAmount;vFieldRef)

$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice date";
                                                    vTableInvoices;vFieldDate)
$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice amount";
                                                  vTableInvoices;vFieldAmount)
$errCode:=OP Get one field number (vConnectID;"[Customers]Ref";
                                                    vTableCustomers;vFieldRef)

   `Compute this year's 1st of january
vValue:="01/01/" + String ( year of ( Current date ) ;"####")

   ` Query for this year's invoices
$errCode:=OP Single query (vConnectID;vTableInvoices;vFieldDate;">";->vValue;
                                                                    vRecords)

   `Order by decreasing amount (largest amount is first of selection)
$errCode:=OP Single order by (vConnectID;vTableInvoices;vFieldAmount;"<")

   ` Focus on the 10 largest invoices
$errCode:=OP Reduce selection (vConnectID;vTableInvoices;10)

   ` Create a selection of customers for those invoices
⇒  $errCode:=OP Many to one join (vConnectID;vTableInvoices;vTableCustomers)

   ` Find out how many customers in the new selection
$errCode := OP Records in selection (vConnectID;vTableCustomers;vRecords)
ALERT (" This year's top ten invoice correspond to " + String (vRecords) + " customers")

PrintCustomers
```

See Also

OP One to many join, OP Records in selection, RELATE ONE SELECTION.

OP One to many join

Selections

version 1.5

OP One to many join (connectionID; oneTableID; manyFieldID) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
oneTableID	Longint	→	Number of the One table in the database
manyFieldID	Longint	→	Number of the Many field in the table
Function result	Longint	←	Error code result for the function

Description

OP One to many join creates a new selection of records in manyTableID based on the selection in the one file.

Based on the example for OP Many to one join example, in the [Customers]/[Invoices]table structure, a call to OP One to many join for [Invoice]Customer ID will select the invoices in the [Invoices] table that refer to the customers in the [Customers] table current selection.

OP Many to one join works with both manual and automatic relations.

Error Codes

If OP One to many join executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9951	Field is not related to another one.
-9971	Field number is out of range.
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

This example finds the list of all last year's invoices received by the customers who got the top 10 invoices of the current year. In other words, if the largest invoice of this year is to ACME Inc., then we want the list of all the invoices to ACME Inc. for last year.

```
C_LONGINT (vRecords; $ErrCode)
C_STRING (10;vValue1;vValue2;vValue3)
C_LONGINT (vTableInvoices;vTableCustomers;$unused1)
C_LONGINT (vFieldDate;vFieldAmount;vFieldCustID;$unused2)
   ` Get [Invoices] tableID and [Invoices]Invoice date fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice date";
   ` Get [Invoices]Invoice amount fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice amount";
   ` Get [Invoices]Customer ID fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Customer ID";
                                                        $unused1;vFieldCustID)
   ` Get [Customers] tableID
$errCode:=OP Get one field number (vConnectID;"[Customers]Ref";
                                                    vTableCustomers;$unused2)

   `Compute last year's 1st of January
vValue1:="01/01/" + String ( year of ( Current date )-1 ;"####")
   `Compute last year's 31st of December
vValue2:="12/31/" + String ( year of ( Current date )-1 ;"####")
   `Compute this year's 1st of January
vValue3:="01/01/" + String ( year of ( Current date ) ;"####")
   ` Query for invoices dated after January 1st last year
$errCode:=OP Single query (vConnectID;vTableInvoices;vFieldDate;">=";
                                                        ->vValue1;vRecords)
   ` Query in the resulting selection fo invoices dated before December 31st last year
$errCode:=OP Single query selection(vConnectID;vTableInvoices;vFieldDate;
                                                    "<=";->vValue2;vRecords)
   `Order by decreasing amount (largest amount is first of selection)
$errCode:=OP Single order by (vConnectID;vTableInvoices;vFieldAmount;"<")
   ` Focus on the 10 largest invoices
$errCode:=OP Reduce selection (vConnectID;vTableInvoices;10)
   ` Create a selection of customers for those invoices
$errCode:=OP Many to one join (vConnectID;vTableInvoices;vTableCustomers)
   ` From that selection of customers, create the selection of all their invoices
⇒  $errCode:=OP One to many join (vConnectID;vTableInvoices;vFieldCustID)
   ` Query the resulting selection for invoices dated afterJanuary 1st this year
$errCode:=OP Single query selection(vConnectID;vTableInvoices;vFieldDate;">="
                                                        ;->vValue3;vRecords)
PrintInvoices
```

See Also

OP Many to one join, OP Single query, RELATE MANY SELECTION.

OP Scan index

version 1.5

OP Scan index (connectionID; tableID; fieldID; selectionSize; scanOrigin) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
fieldID	Longint	→	Number of the field in the table
selectionSize	Longint	→	New size for the target table's current selection
scanOrigin	Longint	→	Origin of scan (0 is end of index)
Function result	Longint	←	Error code result for the function

Description

OP Scan index creates a new selection of selectionSize records for the table specified by taleID. OP Scan index returns a specific number of records from either the top or bottom of the index, depending on the specified direction.

fieldID must be an indexed field. fieldID must be a field from table tableID.

If selectionSize is greater than the number of records in tableID, all the records are selected.

If scanOrigin is less than 0, the records come from the end of the index. If scanOrigin is greater than or equal to 0, the records come from the beginning of the index.

Error Codes

If OP Scan index executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9970	Field is not indexed.
-9971	Field number is out of range.
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

This example creates and prints a selection of this year's invoices out of the top 100 invoices.

```
C_LONGINT(vRecords; $ErrCode)
C_REAL(vValue;vMeanAmount)
C_LONGINT(vTable;vFieldDate;vFieldAmount;vFromEnd)

$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice date";vTable;
                                                                    vFieldDate)
$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice amount";vTable;
                                                                  vFieldAmount)

   ` Scan will be processed from end of index
vFromEnd:= -1
   ` Create a selection of the top 100 invoices
⇒  $errCode := OP Scan index (vConnectID;vTable;vFieldAmount;100;vFromEnd)

   ` Compute this year's 1st of january
vValue:="01/01/" + String ( year of ( Current date ) ;"####")
   ` Query for this year's invoices out of the top 100
$errCode:=OP Single query selection (vConnectID;vTable;vFieldDate;">";
                                                                ->vValue;vRecords)
PrintInvoices
```

See Also

OP All records, OP Order by, OP Reduce selection, OP Single query, SCAN INDEX.

8

Arrays

Arrays, Introduction

version 1.5

The routines described in this chapter allow you to manage data in arrays:

- OP Selection to array - places the current selection of records into an array, which can be represented by pop-up menu(s) or scrollable area(s).
- OP Distinct values - creates an array containing the distinct values of an alphanumeric indexed field.
- OP Array to selection - copies one or more arrays into a selection of records.
- OP Subselection to array - places a portion of the records from the current selection into an array.

OP Selection to array

Arrays

version 1.5

OP Selection to array (connectionID; bindID{; tableID}) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
bindID	Longint	→	Bind list ID
tableID	Longint	→	Number of the table in the database
Function result	Longint	←	Error code result for the function

Description

OP Selection to array places the server fields into arrays. Both the server fields and the arrays are defined in bindID by using OP Define bind by pointer and/or OP Define bind by numbers.

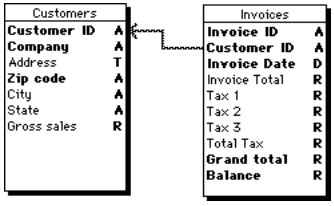
Using OP Selection to array, you can retrieve data from a single target table, or you can retrieve data from a target table and its related tables. To retrieve data from a target file and its related tables:

- The target table must be a Many table and the related table must be a One table.
- The relation must be automatic.
- The tables/fields to be retrieved from the target table and the related table must be in the same bind list.

tableID is the table number of the target table. If you omit this parameter or pass 0, the table number of the first table/field couple in the bind list is used as the target table.

Retrieving records across relations

Consider the following structure:



Case 1

Suppose you want to retrieve the [Invoices]Invoice ID, [Invoices]Invoice Date, [Invoices]Invoice Total, [Customers]Company, and [Customers]City fields, in this order. You retrieve fields from the two tables using a relation, so the target table must be the Many table, in this case [Invoices]. Because the first field is from this table, you just need to build the bind appropriately, and you can omit the target table.

Case 2

Suppose you want to retrieve the [Customers]Company and [Customers]City fields based on the selection in the [Invoices] table. You need to build the bind appropriately, and you have to pass [Invoices] as the target table. Here, you do not retrieve fields from the [Invoices] table even though you use it as the target table.

Error Codes

If OP Selection to array executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
-9969	Invalid field type requested.
-9971	Field number is out of range.
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10135	Invalid parameter type.
10136	The connection does not exist.
10137	The context does not exist.
10139	The context is not defined.
10153	Unable to convert this type of data.
10154	This command cannot be executed right now.

Example

See the example for OP Array to selection .

See Also

OP Define bind by numbers, OP Define bind by pointer, OP Selection to array, SELECTION TO ARRAY.

OP Distinct values

Arrays

version 1.5

OP Distinct values (connectionID; tableID; fieldID; distinctValues) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
fieldID	Longint	→	Number of the (indexed) field in the table
distinctValues	Array	←	Array containing disinct values found in current selection
Function result	Longint	←	Error code result for the function

Description

OP Distinct values fills distinctValues with the unique values from fieldID, which is in tableID, from the current selection.

fieldID must be the number of an indexed Alphanumeric field.

distinctValues is an array of type String, Text, Integer, Long Integer, Real, Date, or Boolean.

This function does not change the current selection or the current record.

Error Codes

If OP Distinct values executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
-9970	Field is not indexed.
-9971	Field number is out of range.
-9972	table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10135	Invalid parameter type.
10136	The connection does not exist.
10153	Unable to convert this type of data.
10154	This command cannot be executed right now.
10158	The field is not an indexed alphanumeric field.

Example

This example presents the user with a list (scrollable area) of today's customers cities.

```
C_LONGINT (vRecords; $ErrCode)
C_STRING (10;vValue)
ARRAY STRING (25; aCities;0)

   `Compute today's date
vValue:= String ( Current date )

` Query for invoices dated of the day, on the [Invoices]Invoice date field (2;3)
$errCode:=OP Single query (vConnectID;2;3;"=";->vValue;vRecords)

   ` Create a selection of customers for those invoices
$errCode:=OP Many to one join (vConnectID;1;2)

   ` Load distinct (no duplicates) city names from [Customers]City (1;5) into the
   ` aCities array
$errCode := OP Distinct values (vConnectID;1;5;aCities)
```

See Also

DISTINCT VALUES, OP Get one field number, OP Selection to array, OP Single query.

OP Array to selection

Arrays

version 1.5

OP Array to selection (connectionID; bindID) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
bindID	Longint	→	Bind list ID
Function result	Longint	←	Error code result for the function

Description

OP Array to selection copies the arrays of the bind list bindID into a selection of records. All of the fields defined by bindID must be in the same table. If there is no current selection this routine has no effect.

If a selection exists, the elements of the array are put into the records based on the order of the array and the order of the records. If there are more elements than there are records, new records are created.

If the arrays are of different sizes, the array with the smallest number of elements is used to determine how many elements will be copied.

Error Codes

If OP Array to selection executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
-9969	Invalid field type requested.
-9971	Field number is out of range.
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10135	Invalid parameter type.
10136	The connection does not exist.
10137	The bind list does not exist.
10138	The bind list is not related to this file.
10139	The bind list is not defined.
10153	Unable to convert this type of data.
10154	This command cannot be executed right now.
10162	The array(s) contain no elements.

Example

This example loads the [Customers]Gross sales into an array, adds the invoice amount of the day to each individual entry, and stores the result back to the [Customers] table.

```
C_LONGINT($i;$ErrCode)
C_LONGINT ($custSalesBind;$invAmountBind)

ARRAY REAL (aCustSales;0)
ARRAY REAL (aInvAmount;0)
ARRAY STRING (10;aCustID;0)
ARRAY STRING (10;aInvCustID;0)

   ` Create a new bind list
$ErrCode := OP Create bind ($custSalesBind)
   ` Add to the list a bind between [Customers]Gross sales (1;7) and aCustSales array
$ErrCode := $ErrCode + OP Define bind by pointer ($custSalesBind;1;7;->aCustSales)
   ` Add to the list a bind between [Customers]Customer ID (1;1) and aCustID array
$ErrCode := $ErrCode + OP Define bind by pointer ($custSalesBind;1;1;->aCustID)

   ` Create a new bind list
$ErrCode := OP Create bind ($invAmountBind)
   ` Add to the list a bind between [Invoices]Total (2;4) and the aInvAmount array
$ErrCode := $ErrCode+OP Define bind by pointer($invAmountBind;2;4;->aInvAmount)
   ` Add to the list a bind between [Invoices]Customer ID (2;2) and aInvCustID array
$ErrCode := $ErrCode + OP Define bind by pointer ($invAmountBind;2;2;->aInvCustID)

C_LONGINT (vRecords; $ErrCode)
C_STRING (10;vValue)

   `Compute today's date
vValue:= String ( Current date )

   ` Query for invoices dated of the day
$errCode:=OP Single query (vConnectID;2;3;"=";->vValue;vRecords)

   ` Load today's invoice amounts into the aInvAmount array
$errCode:= OP Selection to array (vConnectID;$invAmountBind;2)

   ` Create a selection of customers for those invoices
$errCode:=OP Many to one join (vConnectID;1;2)

   ` Load customers gross sales into the aCustSales array
$errCode:= OP Selection to array (vConnectID;$custSalesBind;1)
```

```
   ` For each invoice
For ($i;1; Size of array (aInvAmount) )
      ` Find the customer's entry in the customer ID array
  $cust:= Find in array (aCustID;aInvCustID{$i})
  If ($cust>0)
      ` Accumulate invoice amount into customer gross sales array entry
    aCustSales{$cust} := aCustSales{$cust} + aInvAmount{$i}
  End if
End for

   ` Write back the gross sales values
$errCode:= OP Array to selection (vConnectID;$custSalesBind)
```

See Also

ARRAY TO SELECTION, OP Array to selection, OP Create bind, OP Many to one join, OP Selection to array, OP Single query.

OP Subselection to array

Arrays

version 1.5

OP Subselection to array (connectionID; bindID; firstRecord; lastRecord{; tableID}) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
bindID	Longint	→	Bind list ID
firstRecord	Longint	→	Relative number of first record to retrieve in current selection
lastRecord	Longint	→	Relative number of last record to retrieve in current selection
tableID	Longint	→	Number of the table in the database
Function result	Longint	←	Error code result for the function

Description

OP Subselection to array works exactly like OP Selection to array, except that instead of loading values from all records in the selection, it loads values from records relatively placed in the current selection, between firstRecord and lastRecord.
For a complete discussion of this command functionalities refer to OP Selection to array.

Example

This example exports customer names to a text file. In order to avoid creating large array from a large [Customers] table, it loads the values in 'chunks' of 100 at a time.

```
C_LONGINT ($errCode;vRecords;$custNamesBind)
ARRAY STRING (40;aCustName;0)
   ` Create a new bind list
$ErrCode := OP Create bind ($custNamesBind)
   ` Add to the list a bind between [Customers]Company (1;2) and aCustName array
$ErrCode := $ErrCode +OP Define bind by pointer ($custNamesBind;1;2;->aCustName)
   ` Select all records in the [Customers] table (table 1)
$ErrCode:= OP All records (vConnectID;1)
   ` Get the number of records selected
$errCode := OP Records in selection (vConnectID;1;vRecords)
$i:=0
While ($i<vRecords)  ` Load 100 customers names into the array
⇒     $errCode := OP Subselection to array ( vConnectID; $custNamesBind ; $i+1; $i+100)
      ExportCustNames (->aCustName)
      $i := $i + 100
End while
$errCode := OP Delete bind ($custNamesBind)
```

See Also

OP Define bind by pointer, OP Get one field number, OP Records in selection, OP Selection to array, OP Subselection to array.

9

Transactions

Transactions, Introduction

Transactions are a series of related data modifications that are made to a database within a connection. A transaction is not saved permanently to a database until the transaction is validated. If a transaction is not completed, either because it was cancelled or because of some outside event, the modifications are not saved.

During a transaction, all changes that are made to the data within a connection are stored locally in a temporary buffer. If the transaction is accepted, the changes are saved permanently. If the transaction is cancelled, the changes are not saved.

After a transaction is validated or cancelled, the selection of each table for the current connection becomes empty, because transactions deal with temporary record addresses. For the same reason, you should be cautious when using named selections inside a transaction. After a transaction is validated or cancelled, a named selection created before or during the transaction may contain incorrect record addresses. For instance, a named selection may contain the address of a deleted record or the temporary address of a record added during the transaction.

The following functions manage transactions:

- OP Start transaction - starta a transaction call.
- OP Validate transaction - saves the changes made during the transaction started by OP Start transaction.
- OP Cancel transaction - cancels the changes made.

OP Start transaction (connectionID) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
Function result	Longint	←	Error code result for the function

Description

OP Start transaction starts a transaction for the current connection. All changes to the database will be stored temporarily until the transaction is accepted (validated, commited) or cancelled (rolled back).

For each open connection, you can start only one transaction. You cannot nest transactions. If you start a transaction inside another transaction, 4D Server ignores the second transaction.

Error Codes

If OP Start transaction executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

In this example we are going to add a new invoice. We will also update the gross sales field of the customers table. We will use a transaction to guarantee that either both operations are carried-out or none.

```
C_LONGINT ($BindInvID;$BindInvID;$ErrCode)
C_LONGINT ($transErr)

C_DATE (vInvDate)
C_REAL (vInvTotal;vCustSales)
C_STRING (30;vCustID;vInvoiceID)

C_LONGINT (vTableInvoices;vTableCustomers;vRecords;vIsLocked;$unused1)
C_LONGINT (vFieldAmount;vFieldInvCustID;vFieldInvDate;vFieldInvID;$unused2)
C_LONGINT (vFieldCustID;vFieldCustName;vFieldSales)
```

```
   ` Get [Invoices] tableID and [Invoices]Invoice ID fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice ID";vTableInvoices;
                                                                  vFieldInvID)
   ` Get [Invoices]Customer ID fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Customer ID";$unused1;
                                                                  vFieldInvCustID)
   ` Get [Invoices]Invoice date fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice date";$unused1;
                                                                  vFieldInvDate)
   ` Get [Invoices]Invoice total fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice total";$unused1;
                                                                  vFieldAmount)

   ` Get [Customers] tableID and [Customers]Customer ID fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Customer ID";
                                                  vTableCustomers;vFieldCustID)
   ` Get [Customers]gross sales fieldID
$errCode:=OP Get one field number (vConnectID;"[Customers]Gross sales";
                                                  $unused1;vFieldSales)
   ` Get [Customers]Company fieldID
$errCode:=OP Get one field number (vConnectID;"[Customers]Company";$unused1;
                                                  vFieldCustName)

   `...Connect to the server

   ` Create the bind list to create a new invoice
$ErrCode := OP Create bind ($BindInvID)
$ErrCode := OP Define bind by numbers ($BindInvID;vTableInvoices;vFieldInvID;0;0;
                                                                  "vInvoiceID")
$ErrCode := OP Define bind by numbers ($BindInvID;vTableInvoices;vFieldInvCustID;
                                                                  0;0;"vCustID")
$ErrCode := OP Define bind by numbers ($BindInvID;vTableInvoices;vFieldInvDate;
                                                                  0;0;"vInvDate")
$ErrCode := OP Define bind by numbers ($BindInvID;vTableInvoices;vFieldAmount;
                                                                  0;0;"vInvTotal")

   ` Create the bind list to load and modify a customer record
$errCode := OP Create bind ($BindCustID)
$ErrCode := OP Define bind by numbers ($BindCustID;vTableCustomers;vFieldSales;
                                                                  0;0;"vCustSales")
$ErrCode := OP Define bind by numbers ($BindCustID;vTableCustomers;vFieldCustID;
                                                                  0;0;"vCustID")

   ` Enclose all coming REMOTE database operations in a transaction
⇒  $transErr := OP Start transaction (vConnectID)
```

```
   ` Make the customer record, the current record
vValue:= "ACME Inc."
$errCode := OP Single query (vConnectID;vTableCustomers;vFieldCustName;"=";
                                                              ->vValue;vRecords)

   ` Let's assume the record exist
   ` Set the table to Read/write in order to lock the record
$errCode := OP Set access mode (vConnectID;vTableCustomers;1)

   ` Load the record
Repeat
  vIsLocked := 0
   ` The $BindCustID bind list OP Load record where to put the record data when it
   ` loads it.
  $errCode := OP Load record (vConnectID;$BindCustID;vTableCustomers;vIsLocked)
Until (vIsLocked =0) `assuming it does not remain locked for too long

   ` Create/collect/generate invoice data
vInvoiceID := "0102030405"
vInvDate := Current date
vInvTotal := 100
   ` vCustID has already been set by loading the customers record, thanks to the
   `  $BindCustID bind list

   ` Send the new invoice to the server
$transErr := $transErr + OP New record (vConnectID;$BindInvID)

   ` Let's update the customer record
vCustSales := vCustSales + vInvTotal
$transErr := $transErr +  OP Update record (vConnectID;$BindCustID)

If ($transErr = 0) ` if all went well
      ` Save changes to both tables
⇒     $errCode := OP Validate transaction (vConnectID)
Else
      ` Let's cancel all operations since Start Transaction (New record, Update record)
⇒     $errCode := OP Cancel transaction (vConnectID)
End if

$ErrCode:=OP Delete bind ($BindInvID)
$ErrCode:=OP Delete bind ($BindCustID)
```

See Also

OP Cancel transaction, OP Validate transaction, START TRANSACTION.

OP Validate transaction

Transactions

version 1.5

OP Validate transaction (connectionID) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
Function result	Longint	←	Error code result for the function

Description

OP Validate transaction validates the transaction in the current connection that was started with OP Start transaction. OP Validate transaction saves the changes to the database that occurred during the transaction.

Error Codes

If OP Validate transaction executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

See example for OP Start transaction.

See Also

OP Cancel transaction, OP Start transaction, VALIDATE TRANSACTION.

OP Cancel transaction

OP Cancel transaction (connectionID) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
Function result	Longint	←	Error code result for the function

Description

OP Cancel transaction cancels the transaction in the current connection that was started with OP Start transaction. OP Cancel transaction leaves the database unchanged by cancelling the operations executed during the transaction.

Error Codes

If OP Cancel transaction executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

See example for OP Start transaction.

See Also

CANCEL TRANSACTION, OP Start transaction, OP Validate transaction.

10

Named Selections

Named Selections, Introduction

Named selections offer an easy way to manipulate several selections simultaneously. A named selection is an ordered list of records for a table in a connection. This ordered list of records—along with the order of the records in the selection and the current record—can be given a name and kept in memory.

If you change the selection of records for a table, you can immediately return to a particular selection of records by using a previously created named selection. You can have multiple named selections for each file in a connection.

The following functions are discussed in this section:

- OP Copy named selection - copies the current selection of records into a named selection.
- OP Cut named selection - removes the current selection of records and place them into a named selection.
- OP Use named selection - enables use of a named selection previously created with either OP Copy named selection or OP Cut named selection .
- OP Clear named selection - clears (from memory) a named selection that was previously created with OP Copy named selection.

Process and Interprocess Named Selections

A single user can be connected to the same 4D Server database more than once. Each connection is independent, allowing the user to perform different operations in each. For example, the user could enter a record in one connection and print a series of records in another connection.

When you are creating a named selection, you can decide whether or not the named selection can be shared among all the user's processes.

A shared named selection is called an **interprocess named selection**. Because it is interprocess, any process can use the named selection to recreate the selection of records.

A named selection that is not shared is called a **process named selection**. This type of named selection can be used only by the process in which it was created.

You denote whether a named selection should be process or interprocess through its name. The name of an interprocess named selection must start with the interprocess variable symbol <>. For example, "<>GoodCustomers" denotes an interprocess named selection.

OP Copy named selection (connectionID; tableID; selectionName) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
selectionName	String	→	Name of the selection
Function result	Longint	←	Error code result for the function

Description

OP Copy named selection copies the current selection for tableID to SelectionName. This function does not affect the current selection of tableID.

Error Codes

If OP Copy named selection executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

This example lets the user create a selection of invoices. Then it launches a background process that prints the selection of invoices while the user continues to work with the selection.

```
C_LONGINT($ErrCode;vRecords)

ARRAY LONGINT(arTableID;0)
ARRAY LONGINT(arFieldID;0)

ARRAY TEXT(arLogOp;0)
ARRAY TEXT(arQueOp;0)
ARRAY TEXT(arValues;0)

UserSelectInvoices  `Display a dialog that let the user specify his criteria

   ` Perform the query specified by the user
$errCode := OP Multi query (<>vConnectID;2;arFileID;arFieldID;arLoglOp;arQueOp;
                                                        arValues;vRecords)
```

```
      ` Make a copy of [Invoices] current selection into an Interprocess named selection
⇒     $errCode := OP Copy named selection (<>vConnectID;2;"<>Invoices to print")
      ` Using OP Copy instead of OP Cut preserves the current selection for futher usage
      $printProcess := New process ("PrintInvoices";32000;"Background printing")

      ` Let's display the list of invoice for the user to work with
      UserDisplayInvoices
      ...
```

This is a possible implementation of the 'PrintInvoices' method.

```
      ` Command PrintInvoices Prints in the background the invoices
      ` designated by the "<>Invoices to print" named selection

      ` Make the current selection a 'replica' of the named selection
      $errCode:= OP Use named selection (<>vConnectID;2;"<>Invoices to print")

      If ($errCode=0)

         ` Clear the named selection to release server memory
         $errCode := OP Clear named selection (<>vConnectID;"<>Invoices to print")
         ` Count records in current selection
         $errCode := OP records in selection (<>vConnectID;2;vRecords)

         For ($i;1;vRecords)
            $errCode := OP Goto selected record (<>VconnectID;2;$i)
            PrintOneInvoice (<>vConnectID)
         End for

      End if
```

See Also

OP Clear named selection, OP Cut named selection, OP Use named selection.

OP Cut named selection

Named Selections

version 1.5

OP Cut named selection (connectionID; tableID; selectionName) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
selectionName	String	→	Name of the selection
Function result	Longint	←	Error code result for the function

Description

OP Cut named selection creates a selection named selectionName and moves the current selection for tableID to it. After this routine is called, the selection in tableID becomes empty.

When you use that named selection by calling OP Use named selection, it gets deleted.

Error Codes

If OP Cut named selection executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

This example is very similar to the example for OP Copy named selection. The difference is that it lets the user create several named selections and print them in the background. Since the user will create the selections but not display them, we do not need to use OP Copy named selection to preserve the current selection. Instead, we use OP Cut named selection to avoid wasting server memory.

```
C_LONGINT($ErrCode;vRecords)

ARRAY LONGINT(arTableID;0)
ARRAY LONGINT(arFieldID;0)

ARRAY TEXT(arLogOp;0)
ARRAY TEXT(arQueOp;0)
ARRAY TEXT(arValues;0)

ARRAY STRING (0;<>arPrintRequests;0)

Repeat

   `Display a dialog that let the user specify his criteria
   UserSelectInvoices

   ` Perform the query specified by the user
   $errCode := OP Multi query  (<>vConnectID;2;arFileID;arFieldID;arLoglOp;arQueOp;
                                                              arValues;vRecords)

   ` First get a unique name in the form : <>InvoicesToPrint0001,
   ` <>InvoicesToPrint0002 ...
   $selectionName := "<>InvoicesToPrint" + String (Size of array (<>arPrintRequest)+1;
                                                                        "0000")

   `Append an entry to the print requests array
   INSERT ELEMENT (<>arPrintRequest;Size of array (<>arPrintRequest)+1;1)

   ` Move [Invoices] current selection into an Interprocess named selection using
   ` OP Cut instead of OP Copy in order to empty the current selection saving memory
   ` on the server
⇒  $errCode := OP Cut named selection (<>vConnectID;2;$selectionName)
   <>arPrintRequest{Size of array (<>arPrintRequest)} := $selectionName

Until (UserClickedPrintAndQuit=1)

...
```

This is a possible implementation of the BackgroundPrinter method.

```
` Command BackgroundPrinter Prints in the background the invoices
` designated by the <>arPrintRequest array. Print occur from the bottom.
` process is launched at startup and remains in the background waiting for requests

While (True) ` loop will end on exit

   If (Size of array (<>arPrintRequest)=0)
      `If no print request available go to sleep for 10 secs
       DELAY PROCESS (Current process;60*10)
   Else
       ` Get the name of the selection
      $SelectionName := <>arPrintRequest{1}
      ` Remove it from the queue
      DELETE ELEMENT (<>arPrintRequest;1;1)

      ` Move the named selection at bottom of request queue to the current selection
      ` since named selection was created with a OP Cut calling OP Use will clear the
      ` named selection
      $errCode:= OP Use named selection (<>vConnectID;2;$SelectionName)

      If ($errCode=0)
          ` Count records in current selection
         $errCode := OP records in selection (<>vConnectID;2;vRecords)
         For ($i;1;vRecords)
            $errCode := OP Goto selected record (<>VconnectID;2;$i)
            PrintOneInvoice (<>vConnectID)
         End for
      End if
   End if
End while
```

See Also

OP Clear named selection, OP Copy named selection, OP Single query, OP Use named selection.

OP Use named selection

Named Selections

version 1.5

OP Use named selection (connectionID; tableID; selectionName) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	←	Number of the table in the database
selectionName	String	→	Name of the selection
Function result	Longint	←	Error code result for the function

Description

OP Use named selection makes selectionName the current selection. The table number is returned in tableID.
When you create a named selection, the current record is "remembered" by the named selection. OP Use named selection retrieves the position of the current record and makes it the new current record of the new current selection.

Like the current selection, a named selection does not contain actual records—rather, it contains a list of references to the records. For this reason, if you delete or modify any of the records after you have created the named selection, the following may occur when you reuse the named selection:
• Deleted records will appear as blank records in the new current selection,
• Records added to the same table that have reused the space freed by deleted records may appear in the new current selection,
• Modified records will appear modified in the new current selection.

If the named selection was originally copied, you can call OP Clear named selection to clear the named selection when you are done with it. If the named selection was originally cut, OP Use named selection automatically clears it.

Error Codes

If OP Use named selection executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9977	The named selection does not exist.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

See the examples for OP Copy named selection and OP Cut named selection.

See Also

OP Clear named selection, OP Copy named selection, OP Cut named selection.

OP Clear named selection

Named Selections

version 1.5

OP Clear named selection (connectionID; tableID; selectionName) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
selectionName	String	→	Name of the selection
Function result	Longint	←	Error code result for the function

Description

OP Clear named selection clears selectionName from the server and thus frees the memory it was using. This function does not affect the current selection.

You do not need to clear a named selection that was originally cut and then used.

Error Codes

If OP Clear named selection executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9946*	Unable to clear the named selection because it does not exist.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

See the example for OP Copy named selection.

See Also

OP Copy named selection, OP Cut named selection, OP Use named selection.

11

On a Series

On a Series, Introduction

version 1.5

The routines in this section perform calculations on a series of values in the current selection of records. These routines work on numeric fields only. Numeric fields include Integer, Longint and Real fields.

- OP Sum - calculates the sum of all values in the current selection.
- OP Average - calculates the average of all values in the current selection.
- OP Min - calculates the minimum value in the current selection.
- OP Max - calculates the maximum value in the current selection.

OP Sum

On a Series

version 1.5

OP Sum (connectionID; tableID; fieldID; result) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
fieldID	Longint	→	Number of the field in the table
result	Variable	←	Sum of the values in field for the current selection
Function result	Longint	←	Error code result for the function

Description

OP Sum returns the sum of all the values for fieldID in the current selection in result. If fieldID is an indexed field the index is used to sum the values (i.e. the response time is shorter).

Error Codes

If OP Sum executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9969	Invalid field type requested (field type is not Longint, Integer, Time, or Real).
-9971	Field number is out of range.
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

This example presents the user with a message displaying the amount of the smallest invoice of the day, the largest amount, the average amount and the total sales fo the day.

```
C_LONGINT (vTable;vFieldAmount;vFieldDate;$errCode;vRecords)
C_STRING (10;vValue)
C_REAL (vAverage;vMax;vMin;vSum)

   ` Get table and field IDs
$errCode:=OP Get one field number (vConnectID;"[Invoices]Total";vTable;
                                                              vFieldAmount)
$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice date";vTable;
                                                                 vFieldDate)

   ` Compute today's date
vValue:= String ( Current date )
   ` Make a selection of today's invoices
$errCode:=OP Single query selection (vConnectID;vTable;vFieldDate;"=";»vValue;
                                                                   vRecords)

   ` Get total amount
⇒  $errCode := $ErrCode + OP Sum (vConnectID;vTable;vFieldAmount;vSum)

   ` Get smallest amount
$errCode := $ErrCode + OP Min (vConnectID;vTable;vFieldAmount;vMin)

   ` Get largest amount
$errCode := $ErrCode + OP Max (vConnectID;vTable;vFieldAmount;vMax)

   ` Get average amount
$errCode := $ErrCode + OP Average (vConnectID;vTable;vFieldAmount;vAverage)

If ($ErrCode=0)
   $mes := " Today' results are " + Char (Carriage return)
   $mes := $mes + "   smallest invoice : " + String (vMin;"### ###.00 ) +
                                                        Char (Carriage return)
   $mes := $mes + "   largest invoice   : " + String (vMax;"### ###.00 ) +
                                                        Char (Carriage return)
   $mes := $mes + "   average invoice  : " + String (vAverage;"### ###.00 ) +
                                                        Char (Carriage return)
   $mes := $mes + "   total invoiced     : " + String (vSum;"### ###.00 )
   ALERT ($mes)
End if
```

See Also

OP Average, OP Max, OP Min.

OP Average

On a Series

version 1.5

OP Average (connectionID; tableID; fieldID; result) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
fieldID	Longint	→	Number of the field in the table
result	Variable	←	Average value in field for the current selection
Function result	Longint	←	Error code result for the function

Description

OP Average returns the mean value of all the values for fieldID in the current selection in result. If fieldID is an indexed field, the index is used to compute the mean value (i.e., the response time is shorter).

Error Codes

If OP Average executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9969	Invalid field type requested (field type is not Longint, Integer, Time or Real).
-9971	Field number is out of range.
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

See example for OP Sum.

See Also

OP Max, OP Min, OP Sum.

OP Min

On a Series

version 1.5

OP Min (connectionID; tableID; fieldID; result) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
fieldID	Longint	→	Number of the field in the table
result	Variable	←	Minimun value in field for the current selection
Function result	Longint	←	Error code result for the function

Description

OP Min returns the smallest of all the values for fieldID in the current selection in result. If fieldID is an indexed field, the index is used to find the smallest value (i.e., the response time is shorter).

Error Codes

If OP Min executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9969	Invalid field type requested (field type is not Longint, Integer, Time or Real).
-9971	Field number is out of range.
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

See example for OP Sum.

See Also

OP Average, OP Max, OP Sum.

OP Max (connectionID; tableID; fieldID; result) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
fieldID	Longint	→	Number of the field in the table
result	Variable	←	Maximum value in field for the current selection
Function result	Longint	←	Error code result for the function

Description

OP Max returns the largest of all the values for fieldID in the current selection in result. If fieldID is an indexed field, the index is used to find the largest value (i.e., the response time is shorter).

Error Codes

If OP Max executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9969	Invalid field type requested (field type is not Longint, Integer, Time or Real).
-9971	Field number is out of range.
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Examples

See example for OP Sum.

See Also

OP Average, OP Min, OP Sum.

12

Binds

Binds, Introduction

version 1.5

A bind is a link that you establish between a table/field—called a "couple"—on the server and a local table/field, variable, or array.

To modify a record, create a new record or load the data from a current record, you create a bind and establish a link between the server fields and the local fields or variables in which the data will be displayed and/or modified.

To retrieve or send data from the 4D Server database by using arrays, you create a bind and establish a link between the server fields and the local arrays that will contain the data.

The routines that you use for data manipulation depend on whether your binds involve fields, variables or arrays. The following table describes which routines can be used with each type of 4th Dimension object.

Objects	Routines
Variables or fields	OP New record
	OP Update record
	OP Load record
	OP Goto record
	OP Goto selected record
Arrays	OP Selection to array
	OP Array to selection
	OP Subselection to array
	and the Records commands (listed in the Records section)

The following routines allow you to create, define and delete binds:

- OP Create bind creates an empty bind list in memory.
- OP Define bind by numbers and/or OP Define bind by pointer - define a bind that has been created with OP Create bind.

Note: You can use OP Define bind by numbers or OP Define bind by pointer on the same bind.

- OP Delete bind - allows you to free the memory used by a bind created with OP Create bind.
- OP Set format - sets the format used when non-alphanumeric values are converted to alphanumeric values and vice versa.

OP Create bind

version 1.5

OP Create bind (bindID) → Longint

Parameter	Type		Description
bindID	Longint	←	New bind list ID
Function result	Longint	←	Error code result for the function

Description

OP Create bind creates an empty bind list in memory and returns its bind reference number in bindID.

You add binds to an existing bind list by using OP Define bind by numbers or OP Define bind by pointer.

During data manipulation, pass the reference number of the bind list to indicate which server objects are associated with which local objects.

When you have finished using the bind list, call OP Delete bind to release the memory used by the bind list.

See the examples for OP Define bind by numbers and OP Define bind by pointer.

Error Codes

If OP Create bind executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
10128	The 4D Open for 4th Dimension package has not been initialized.
10154	This command cannot be executed right now.

See Also

OP Define bind by numbers, OP Define bind by pointer, OP Delete bind.

OP Delete bind

Binds

version 1.5

OP Delete bind (bindID) → Longint

Parameter	Type		Description
bindID	Longint	→	Bind list ID
Function result	Longint	←	Error code result for the function

Description

OP Delete bind releases the memory used by bindID bind list. Use this function only after you have finished using bindID.

Error Codes

If OP Delete bind executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
10128	The 4D Open for 4th Dimension package has not been initialized.
10137	The bind list does not exist.
10154	This command cannot be executed right now.

See Also

OP Create bind, OP Define bind by numbers, OP Define bind by pointer.

OP Define bind by numbers

Binds

version 1.5

OP Define bind by numbers (bindID; tableID; fieldID; localTableID; localFieldID; variableName) → Longint

Parameter	Type		Description
bindID	Longint	→	Bind list ID
tableID	Longint	→	Number of table in the database
fieldID	Longint	→	Number of the field in the table
localTableID	Longint	→	Number of the table in the local database (0 if bind to variable)
localFieldID	Longint	→	Number of the field in the local table (0 if bind to variable)
variableName	String	→	Name of the variable or array ("" if bind to field)
Function result	Longint	←	Error code result for the function

Description

OP Define bind by numbers adds a bind to the bindID bind list for tableID and fieldID on the server, indicated by passing their numbers.

For example, to bind the [Invoices]Invoice ID, and Invoice ID is the first field in the [Invoices] table, which is the second file on the server database, pass 1 as fieldID and 2 as tableID.

In the same way, if you bind a field from the server to a local field, pass the numbers of the table and field to localTableID and to localFieldID and "" to variableName.

To bind tableID and fieldID to a local object such as a variable or an array, pass its name to variableName and pass 0 to localTableID and localFieldID.

variableName can be an interprocess or process variable or array name. variableName cannot be a local variable or array (i.e., using the dollar sign "$").

Error Codes

If OP Define bind by numbers executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
10128	The 4D Open for 4th Dimension package has not been initialized.
10137	The bind list does not exist.
10140	The local field does not exist.

Example

This example lists binds between the [Invoices] table fields and several arrays to be used as scrollable areas on a form. This would be a way to display a list of invoices to the user.

```
C_LONGINT ($BindID;$ErrCode;$ConnectID)
ARRAY DATE (aInvDate;0)
ARRAY REAL (aInvTotal;0)
ARRAY STRING (30;aCity;0)
ARRAY STRING (30;aCustID;0)
ARRAY STRING (30;aCompany;0)
C_LONGINT (vTableInvoices;vTableCustomers;$unused1)
C_LONGINT (vFieldDate;vFieldAmount;vFieldCustID;$unused2)
C_LONGINT (vFieldCompany;vFieldCity)

   ` Get [Invoices] tableID and [Invoices]Invoice date fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice  date";
                                                    vTableInvoices;vFieldDate)
   ` Get [Invoices]Invoice amount fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice  amount";
                                                      $unused1;vFieldAmount)
   ` Get [Invoices]Customer ID fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Customer  ID";$unused1;
                                                          vFieldCustID)
   ` Get [Customers] tableID and [Customers]Company fieldID
$errCode:=OP Get one field number (vConnectID;"[Customers]Company";
                                                 vTableCustomers;vFieldCompany)
   ` Get [Customers]City fieldID
$errCode:=OP Get one field number (vConnectID;"[Customers]Company";
                                                      $unused1;vFieldCity)
   ` Connect to the server, perform a query on invoices to create a selection of records
$ErrCode := OP Create bind ($BindID)
⇒  $ErrCode := $ErrCode + OP Define bind by numbers ($BindID;vTableInvoices;
                                                    vFieldCustID;0;0;"aCustID")
⇒  $ErrCode := $ErrCode + OP Define bind by numbers ($BindID;vTableInvoices;
                                                   vFieldCustID;0;0;"aInvDate")
⇒  $ErrCode := $ErrCode + OP Define bind by numbers ($BindID;vTableInvoices;
                                                 vFieldAmount;0;0;"aInvTotal")
⇒  $ErrCode := $ErrCode + OP Define bind by numbers ($BindID;vTableCustomers;
                                                    vFieldCity;0;0;"aCity")
⇒  $ErrCode := $ErrCode + OP Define bind by numbers ($BindID;vTableCustomers;
                                                 vFieldCompany;0;0;"aCompany")
   ` On a known stable structure one could also write
   ` $ErrCode := $ErrCode+OP Define bind by numbers ($BindID;1;2;0;0;"aCompany")
If ($ErrCode=0)
  $ErrCode:=OP Selection to array ($ConnectID;$BindID)
End if
$ErrCode:=OP Delete bind ($BindID)
```

OP Define bind by pointer

Binds

version 1.5

OP Define bind by pointer (bindID; tableID; fieldID; boundObject) → Longint

Parameter	Type		Description
bindID	Longint	→	Bind list ID
tableID	Longint	→	Number of the table in the database
fieldID	Longint	→	Number of the field if the table
boundObject	Pointer	→	Pointer on object to bind
Function result	Longint	←	Error code result for the function

Description

OP Define bind by pointer is similar to OP Define bind by numbers. However, OP Define bind by pointer takes a pointer to the local object: field, variable, or array. This syntax allows for more flexibility when one does not know in advance what local objects are going to be bound.

OP Define bind by pointer adds a bind to bindID bind list for tableID and fieldID on the server, indicated by passing their numbers.

In boundObject, pass a pointer to a field in the local database, to a variable or to an array. For example, you can pass ->[AFile]AField,->Var, -><>Var,->Array1, or -><>Array1. Variables or arrays can be process or interprocess only.

Error Codes

If OP Define bind by pointer executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
10128	The 4D Open for 4th Dimension package has not been initialized.
10137	The bind list does not exist.
10140	The local field does not exist.
10141	The local table does not exist.
10142	You cannot bind this type of field.
10144	The 4D pointer is equal to NIL.
10145	A 4D pointer was expected.
10154	This command cannot be executed right now.
10157	Duplicated bind entry.

Example

This example is essentially the same as the example for OP Define bind by numbers, except that it will carries out the definitions using pointers rather than variable names.

```
C_LONGINT ($BindID;$ErrCode;$ConnectID)
ARRAY DATE (aInvDate;0)
ARRAY REAL (aInvTotal;0)
ARRAY STRING (30;aCity;0)
ARRAY STRING (30;aCustID;0)
ARRAY STRING (30;aCompany;0)
C_LONGINT (vTableInvoices;vTableCustomers;$unused1)
C_LONGINT (vFieldDate;vFieldAmount;vFieldCustID;$unused2)
C_LONGINT (vFieldCompany;vFieldCity)
   ` Get [Invoices] tableID and [Invoices]Invoice date fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice  date";
                                                    vTableInvoices;vFieldDate)
   ` Get [Invoices]Invoice amount fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice  amount";
                                                      $unused1;vFieldAmount)
   ` Get [Invoices]Customer ID fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Customer  ID";$unused1;
                                                          vFieldCustID)
   ` Get [Customers] tableID and [Customers]Company fieldID
$errCode:=OP Get one field number (vConnectID;"[Customers]Company";
                                                 vTableCustomers;vFieldCompany)
   ` Get [Customers]City fieldID
$errCode:=OP Get one field number (vConnectID;"[Customers]Company";
                                                      $unused1;vFieldCity)
   `Connect to the server, perform a query on invoices to create a selection of records
$ErrCode := OP Create bind ($BindID)
⇒  $ErrCode := $ErrCode + OP Define bind by pointer ($BindID;vTableInvoices;
                                                    vFieldCustID;->aCustID)
⇒  $ErrCode := $ErrCode + OP Define bind by pointer ($BindID;vTableInvoices;
                                                    vFieldCustID;->aInvDate)
⇒  $ErrCode := $ErrCode + OP Define bind by pointer ($BindID;vTableInvoices;
                                                   vFieldAmount;->aInvTotal)
⇒  $ErrCode := $ErrCode + OP Define bind by pointer ($BindID;vTableCustomers;
                                                      vFieldCity;->aCity)
⇒  $ErrCode := $ErrCode + OP Define bind by pointer ($BindID;vTableCustomers;
                                                 vFieldCompany;->aCompany)
   ` On a known stable structure one could also write
   ` $ErrCode := $ErrCode + OP Define bind by pointer ($BindID;1;2;0;0;->aCompany)
If ($ErrCode=0)
  $ErrCode:=OP Selection to array ($ConnectID;$BindID)
End if
$ErrCode:=OP Delete bind ($BindID)
```

OP Set format

version 1.5

OP Set format (bindID; tableID; fieldID; conversionFormat) → Longint

Parameter	Type		Description
bindID	Longint	→	Bind list ID
tableID	Longint	→	Number of the table in the database
fieldID	Longint	→	Number of the field in the table
conversionFormat	String	→	Format string for conversions
Function result	Longint	←	Error code result for the function

Description

OP Set format allows you to set the format used when you convert non-alphanumeric values to alphanumeric and text values, and vice versa.

To clear a format, pass an empty string in the conversionFormat parameter.

This command accepts two syntactical forms:
- Pass a valid bind list number in bindID
- Pass a negative number that denotes a data type

If you pass a valid bind list number in bindID, you also pass the numbers of the table and field whose formats you want to set. The format applies only to the field in this bind.

If you pass a negative number, tableID and fieldID are ignored; you can pass 0 for those parameters. The format applies to all conversions that do not have a format already set.

The following table lists the data type codes and some examples of the formats you can use:

Data Type	Data Type Code	Example Format
Numeric	-1	$###,###.00
Date	-4	Char(3) (where 3 is the number of the date format)
Boolean	-6	Male;Female
Time	-11	Char(4) (where 4 is the number of the time format)

For a complete description of valid data types and conversion format, refer to the 4D command SET FORMAT.

Error Codes

If OP Set format executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this command.
10128	The 4D Open for 4th Dimension package has not been initialized.
10137	The bind list does not exist.
10148	Unknown option requested to 4D Open for 4th Dimension.
10154	This command cannot be executed right now.
10163	Bind entry not found.

Example

This example binds a string array to a date field on the server. We use the format comment to specify how the conversion should occur.

```
C_LONGINT ($BindID;$ErrCode;$ConnectID)
ARRAY STRING (35;aInvDate;0)
C_LONGINT (vTableInvoices)
C_LONGINT (vFieldDate)

   ` Get [Invoices] tableID and [Invoices]Invoice date fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice  date";
                                                  vTableInvoices;vFieldDate)

$ErrCode := OP Create bind ($BindID)
$ErrCode := $ErrCode + OP Define bind by pointer ($BindID;vTableInvoices;
                                                  vFieldDate;->aInvDate)
   ` On a known stable structure one could also write
   ` $ErrCode := $ErrCode + OP Define bind by pointer ($BindID;2;3;0;0;->aCompany)

   ` Specify (non default) conversion format for use when transfering data with THIS
   ` bind. We want invoices date read through this bind to be converted to string
   ` using the Abbreviated ,6, or Abbr Month Date
⇒  $errCode:= OP Set format (vConnectID;$BindID;vTableInvoices;vFieldDate;
                                                  Char (Abbr Month Date))

If ($ErrCode=0)
   $ErrCode:=OP Selection to array (vConnectID;$BindID)
End if

$ErrCode:=OP Delete bind ($BindID)
```

See Also

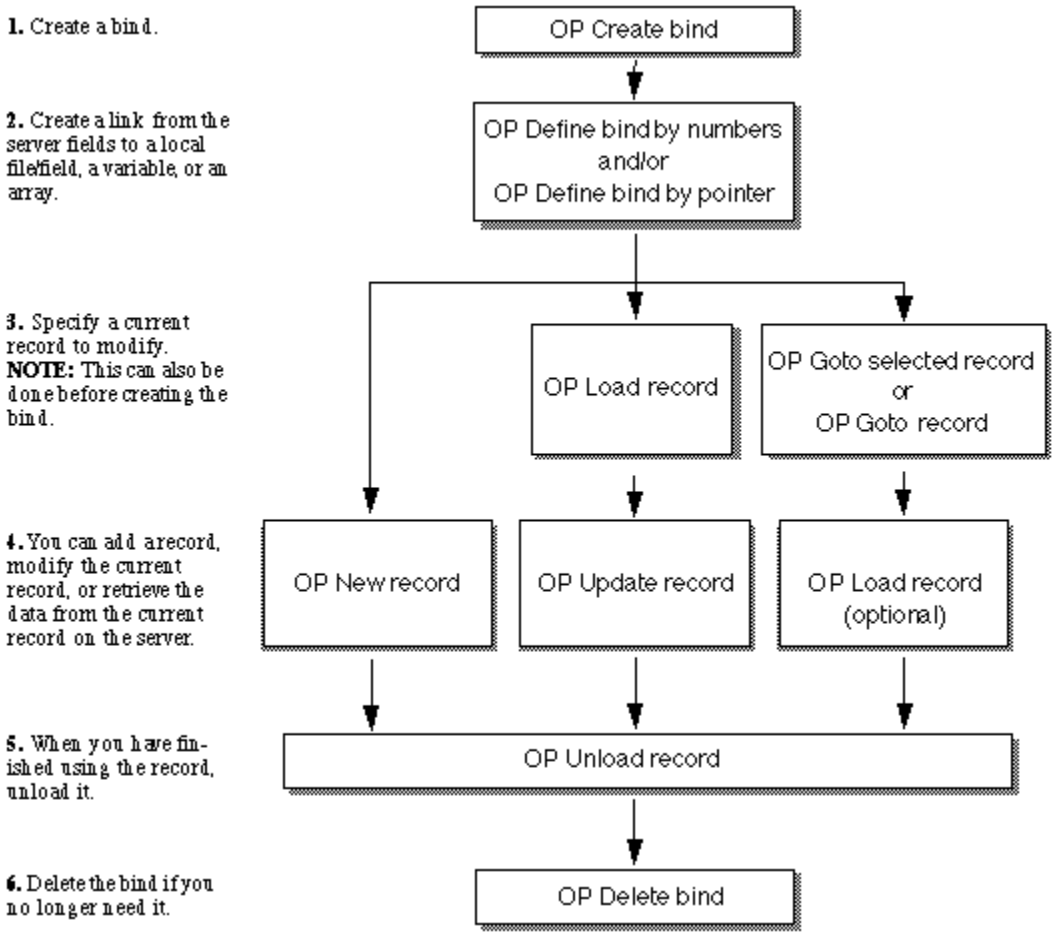
OP Array to selection, OP Define bind by numbers, OP Define bind by pointer, OP Selection to array, OP Set format.

13

Records

To modify a record, create a new record, or load the data from a current record, you create a bind and establish a link between the server fields and the local fields or variables in which the data will be displayed and/or modified.

The following graphic describes the steps involved in creating and defining a bind so that you can eventually manipulate the data from the 4D Server database.



The following routines allow you to manipulate records:

• OP Set access mode - sets the access mode of a table by setting a table to read/write or to read-only mode.

• OP Goto selected record - goes to a specified record in the current selection and makes it the current record. This function optionally loads the field data.

• OP Load record - after you have set a current record by using OP Goto selected record or OP Goto record or by querying, this loads the field data with the fields that you have defined in your bind.

• OP Unload record - unloads a record.

• OP Update record - to modify the current record with the fields that you have defined in your bind, specify the data to use and call this routine.

• OP New record - to create a new record with the fields that you have defined in your bind, specify the data to use and call this routine.

• OP Sequence number - to assign a sequence number to a new record, assign the number returned by this routine.

• OP Delete record - deletes the current record.

• OP Goto record - makes a record the current and only record in the selection.

• OP Get record numbers - gets the record numbers for the current selection of records. The numbers returned can be passed to OP Goto record to make the specified record the current and only one in the selection. This function optionally loads the field data.

• OP Current record number - returns the number of the current record.

OP Set access mode

Records

version 1.5

OP Set access mode (connectionID; tableID; accessMode) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database 0 = all tables
accessMode	Longint	→	Access mode 0 = read-only Otherwise, read-write
Function result	Longint	←	Error code result for the function

Description

OP Set access mode sets the access mode for tableID to either read-only or read/write for this connection.

If you pass 0 in tableID, the function sets the access mode for all the tables in the database.

In read-only mode, the records in tableID that you retrieve on your local machine by using binds are locked. This means that you cannot delete or update those records. You can, however, add records.

When a new connection is opened, all the tables of the database are in read/write mode for this connection.

Read-Only Mode

While a table is in read-only mode, you will be able to display or print records in the table but not modify them. If you just want to display records, set the file to read-only mode. This ensures that other users will be able to access records in the table in read/write mode.

Note that read-only mode only applies to editing existing records. It does not affect the creation of new records. You can add records to a read-only table.

Read/Write Mode

If you want to modify records in a table, the table must be in read-write mode. By default, all tables are in read/write mode when you open a connection.

In read/write mode, you can save a modified record as long as no other user or process has acquired a lock on that record first. If the record is locked, you will be able to view the record but not modify it.

Accessing a table in read/write mode does not mean that a particular record is in read/write mode. In fact, a record might already be in use by another client; in this case, the record is returned to you in read-only mode and it is said to be locked.

To test whether or not a record is locked, you must download at least one field from the record. Three routines allow you to test the lock status of a record: OP Goto selected record, OP Goto record, and OP Load record.

Error Codes

If OP Set access mode executes correctly, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

In this example, we create a selection of customer records in order to print them. Since we do not intend to modify the content of these records, we set the access mode of the [Customers] table to read/only, allowing other processes access to the records.

```
C_LONGINT (vTableID)
C_LONGINT (vFieldCompany;vCompValue)
C_LONGINT ($errCode;vRecords)

   ` Get tabel [Customers] ID and field [Customers]Company ID
$errCode := OP Get one field number (vConnectID;"[Customers]Company";vTableID;
                                                                vFieldCompany)

   ` Query the table for all company with a name begining with Ab
vCompValue := "Ab@"
$errCode := OP Single query (vConnectID;vTableID;vFieldCompany;"=";->vCompValue;
                                                                    vRecords)

   ` Order the selection by Company name
$errCode := OP Single order by (vConnectID;vTableID;vFieldCompany;">")

   ` Set access mode to read only, setting it is only necessary BEFORE loading a record
⇒  $errCode := OP Set access mode (vConnectID;vTableID;0)

   PrintCustomerRecords
```

See Also

OP Load record, OP Unload record, READ ONLY, READ WRITE.

OP Goto selected record

Records

version 1.5

OP Goto selected record (connectionID; tableID; recordNumber{; bindID{; lockStatus}}) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
recordNumber	Longint	→	Relative position of record in the current selection
bindID	Longint	→	Bind list ID
lockStatus	Longint	←	Lock status = 1 means the record is locked
Function result	Longint	←	Error code result for the function

Description

OP Goto selected record moves the current record pointer to the specified record in the current selection in tableID and makes that record the current record. The current selection for tableID is not altered.

If you pass 0 in bindID or omit the last two parameters, OP Goto selected record changes only the current record pointer. If you pass a valid bind list ID, the function downloads the fields accordingly. In this case, if the access mode to the table is read/write, the function will also try to give you access to the record in read/write mode. If this operation is successful, the function returns 0 in lockStatus. If the record is already in use by another process, the function returns 1 in lockStatus.

By testing the lockStatus parameter, you know whether or not you can subsequently modify or delete the record using OP Update record or OP Delete record.

If you obtained the current record in read/write mode, after you are done with the record (unless you deleted it), you must release it for the other processes by using OP Unload record.

Error Codes

If OP Goto selected record executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9968	Invalid selected record number requested.
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

This example implements the PrintCustomerRecords method used in other examples. This method prints all records in the current selection of the [Customers] table.

```
C_LONGINT (vTable;$unused1)
C_LONGINT (vFieldID;vFieldCompany;vFieldAddress;vFieldZip;vFieldCity;vFieldState)
C_STRING (10;vCustID)
C_STRING (30;vCustName;vCustCity)
C_STRING (5;vCustZip)
C_STRING (2;vCustState)
C_TEXT (vCustAddress)
C_LONGINT (vRecords;lockStatus)

   ` Get [Customers] tableID and [Customers]Customer ID fieldID
$errCode:=OP Get one field number (vConnID;"[Customers]Customer ID";
                                                                  vTable;vFieldID)
   ` Get [Customers]Company fieldID
$errCode:=OP Get one field number (vConnID;"[Customers]Company";
                                                        $unused1;vFieldCompany)
   ` Get [Customers]Address fieldID
$errCode:=OP Get one field number (vConnID;"[Customers]Address";
                                                          $unused1;vFieldAddress)
   ` Get [Customers]Zip code fieldID
$errCode:=OP Get one field number (vConnID;"[Customers]Zip code";$unused1;
                                                                  vFieldZip)
   ` Get [Customers]City fieldID
$errCode:=OP Get one field number (vConnID;"[Customers]City";$unused1;vFieldCity)
   ` Get [Customers]State fieldID
$errCode:=OP Get one field number (vConnID;"[Customers]State";
                                                        $unused1;vFieldState)
```

```
   ` Define the bind list that we will use to load the records to print
$ErrCode := OP Create bind ($BindID)
$ErrCode := $ErrCode + OP Define bind by pointer ($BindID;vTable;vFieldID;->vCustID)
$ErrCode := $ErrCode + OP Define bind by pointer ($BindID;vTable;
                                                  vFieldCompany;->vCustName)
$ErrCode := $ErrCode + OP Define bind by pointer ($BindID;vTable;
                                                  vFieldAddress;->vCustAddress)
$ErrCode := $ErrCode + OP Define bind by pointer ($BindID;vTable;
                                                  vFieldZip;->vCustZip)
$ErrCode := $ErrCode + OP Define bind by pointer ($BindID;vTable;
                                                  vFieldCity;->vCustCity)
$ErrCode := $ErrCode + OP Define bind by pointer ($BindID;vTable;
                                                  vFieldState;->vCustState)

   ` The CustRecords form contains the variable that will be loaded through the bind list
OUTPUT FORM ([Dialogs];"CustRecords")
$i := 1
   ` Get the number of records in the current selection of [Customers]
$errCode := OP Records in selection (vConnID;vTable;vRecords)

While ($i <= vRecords)
⇒     $errCode := OP Goto selected record (vConnID;vTable;$i;$BindID;LockStatus)
      `Values are loaded, print them with the print record command
   PRINT RECORD ([Dialogs])
   $i := $i +1
End while

$errCode := OP Delete bind ($BindID)
```

See Also

OP Get record numbers, OP Goto record.

OP Load record

version 1.5

OP Load record (connectionID; bindID; tableID; lockStatus) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
bindID	Longint	→	Bind list ID
tableID	Longint	→	Number of the table in the database
lockStatus	Longint	←	Lock status = 1 means the record is locked.
Function result	Longint	←	Error code result for the function

Description

OP Load record loads and tries to lock the current record(s) of the table(s) listed in the bind list. It loads only the data defined in bindID.

For example, if you bind only the first three fields in a table, only the data from these three fields will be loaded, instead of loading all the fields for that record.

If lockStatus returns 0, this means that you have obtained the record(s) in read/write mode and can modify or delete the record(s). If you obtain a record in read/write mode, you must later release it using OP Unload record. This ensures that other users and processes will be able to access the record in read/write mode.

If the record is already in use, or if you accessed the table(s) in read-only mode, lockStatus = 1. If you need to access the current record in read/write mode, you must re-execute OP Load record until lockStatus returns 0 or until you choose to abort the operation you wanted to perform.

OP Load record always returns the values of the fields you requested, regardless of the access mode.

If there is no current record, OP Load record has no effect.

Error Codes

If OP Load record executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
-9969	Invalid field type requested.
-9971	Field number is out of range.
-9972	Table number is out of range.

4004	There is no current record in the local table.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10137	The bind list does not exist.
10139	The bind list is not defined.
10153	Unable to convert this type of data.
10154	This command cannot be executed right now.

Example

In this example, the LoadAndLock method attempts to load a record and lock it in order to update it later.

```
      ` Method LoadAndLock (Conn,TableID;BindID;MaxWait) -> LockStatus
      ` Conn, Connection ID
      ` TableID, number of the table for whitch to load the current record
      ` BindID, bind list ID to be used for loading values
      ` MaxWait, Maximum nuber of lock attempts before returning failure

C_LONGINT (Conn,TableID,BindID,MaxWait)
C_LONGINT ($errCode;Lock;$0;$i)

$Conn := $1
$TableID := $2
$BindID := $3
$MaxWait := $4
$i := 0
$Lock:=0

   ` Set access mode to read write so that next load operation will attempt to lock the
   ` record
$errCode := OP Set access mode (Conn;TableID;1)

Repeat
⇒     $errCode:= OP Load record ($Conn;$TableID;$BindID;$Lock)
   If ( ($i<=$maxWait) & ($lock=1) )
      ` If the record is locked by another process, and we have some attempts left
      DELAY PROCESS (Current process;15) `Sleep 1/4 of a second
   End if
   $ i:= $i + 1
Until ( ($i >=$maxWait) | ($lock = 0) )

` Return lock status
$0 := $lock
```

See Also

OP Goto selected record, OP Set access mode, OP Unload record.

OP Unload record

Records

version 1.5

OP Unload record (connectionID; tableID) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
Function result	Longint	←	Error code result for the function

Description

OP Unload record releases the current record in tableID, which was returned in read/write mode by OP Load record, OP Goto selected record or OP Goto record.

If there is no current record, OP Unload record has no effect.

Error Codes

If OP Unload record executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

See Also

OP Load record, OP Set access mode, OP Update record, UNLOAD RECORD.

OP Update record

Records

version 1.5

OP Update record (connectionID; bindID) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
bindID	Longint	→	Bind list ID
Function result	Longint	←	Error code result for the function

Description

OP Update record saves the changes made to a record that you have bound. Only the fields defined in bindID are modified.

You must first specify a current record (by using OP Goto selected record or OP Goto record), load it with OP Load record if necessary, assign new values to the fields you want to modify, and then call OP Update record to save the changes.

After loading the record, remember to check whether or not it is locked.

The bind list that you use to test whether or not a record is locked and the bind list that you use to update the record can be different. For example, you may use a one-field bind for testing the lock status of a record, and then use a ten-field bind list for updating the fields of the record.

If the record is locked, the record is not updated and OP Update record does not return an error. You must check the lockStatus parameter of the OP Load record, OP Goto record or OP Goto selected record function.

Error Codes

If OP Update record executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
-9967	The record was not modified because it could not be loaded.
-9969	Invalid field type requested.
-9971	Field number is out of range.
-9972	table number is out of range.
-9998	Duplicate index key.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10137	The bind list does not exist.

10138	The bind list is not related to this table.
10139	The bind list is not defined.
10153	Unable to convert this type of data.
10154	This command cannot be executed right now.

Example

This example updates the gross sales field of a customer's record to reflect the day's invoices.

```
C_LONGINT ($BindInvID;$ErrCode)
C_LONGINT ($transErr;$lock;vRecords)

C_STRING (8;vInvDate)
C_REAL (vInvTotal;vCustSales)
C_STRING (30;vCustID)

C_LONGINT (vTableInvoices;vTableCustomers;vRecords;vIsLocked;$unused1)
C_LONGINT (vFieldAmount;vFieldInvCustID;vFieldInvDate;vFieldInvID;$unused2)
C_LONGINT (vFieldCustID;vFieldSales)

ARRAY STRING (30;CustIDs;0)
ARRAY STRING (30;InvCustIDs;0)
ARRAY REAL (addToGrossSales;0)
ARRAY REAL (InvAmounts;0)

   ` Get [Invoices] tableID and [Invoices]Customer ID fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Customer ID";
                                                    vTableInvoices;vFieldInvCustID)
   ` Get [Invoices]Invoice date fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice date";
                                                        $unused1;vFieldInvDate)
   ` Get [Invoices]Invoice total fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Invoice total";
                                                        $unused1;vFieldAmount)

   ` Get [Customers] tableID and [Customers]Customer ID fieldID
$errCode:=OP Get one field number (vConnectID;"[Invoices]Customer ID";
                                                    vTableCustomers;vFieldCustID)
   ` Get [Customers]gross sales fieldID
$errCode:=OP Get one field number (vConnectID;"[Customers]Gross sales";$unused1;
                                                                vFieldSales)
   ` Get [Customers]Company fieldID
$errCode:=OP Get one field number (vConnectID;"[Customers]Company";
                                                        $unused1;vFieldCustName)
   ` Create the bind list to load and modify a customer record
$errCode := OP Create bind ($BindCustID)
$ErrCode := OP Define bind by pointer ($BindCustID;vTableCustomers;
                                                        vFieldSales;->vCustSales)
```

```
$ErrCode := OP Define bind by pointer ($BindCustID;vTableCustomers;
                                                    vFieldCustID;->vCustID)

   ` Create the bind list to load invoices amounts and customer IDs
$errCode := OP Create bind ($BindInvID)
$ErrCode := OP Define bind by pointer ($BindInvID;vTableInvoices;
                                                    vFieldAmount;->InvAmounts)
$ErrCode := OP Define bind by pointer ($BindInvID;vTableInvoices;
                                                    vFieldInvCustID;->InvCustIDs)

   ` Make the day's invoices the current selection
vInvDate:= String (Current date)
$errCode := OP Single query (vConnectID;vTableInvoices;vFieldInvDate;
                                                    "=";->vInvDate;vRecords)

   ` Load the invoices values in an array, along with the customer IDs.
$errCode := OP Selection to array (vConnectID;$BindInvID)
   ` Make an array of UNIQUE customer's ID to be updated
$errCode := OP Distinct values (vConnectID;vTableInvoices;vFieldInvCustID;CustIDs)
   ` Make the day's sales the same size
ARRAY REAL (addToGrossSales;Size of array(CustIDs))

   ` Loop through the invoices array to cumulate invoice amounts into unique
   ` customer's gross sales entry
For ($i;1; Size of array (InvCustIDs) )
   $uniqCustEntry := Find in array (CustIDs;InvCustIDs{$i})
   addToGrossSales{uniqCustEntry}:=addToGrossSales{uniqCustEntry}+InvAmounts{$i}
End for

   ` Create a selection of customers for those invoices
$errCode:=OP Many to one join (vConnectID;vTableInvoices;vTableCustomers)

   ` Find out how many records were found, even though we already know from the
   ` size of the CustIDs array.
$errCode := OP Records in selection (vConnectID;vTableCustomers;vRecords)

   ` Set the table to Read/write in order to lock the record
$errCode := OP Set access mode (vConnectID;vTableCustomers;1)

   ` Browse through all customers records
For ($i;1; vRecords)
   $errCode := OP Goto selected record (vConnectID;vTableCustomers;$i;
                                                    BindCustID;$lock)
      ` Let's assume the customer record is safely loaded and locked
      ` The OP Goto selected record command was also used to load the record values
```

```
            ` Find the entry for the current customer record in the addToGrossSales array
         $uniqCustEntry := Find in array (CustIDs;vCustID)

            ` Accumulate into the existing gross sales value
         vCustSales := addToGrossSales{$uniqCustEntry}

            ` UPDATE the customer record
⇒        $errCode := OP Update record (vConnectID;$BindCustID)
      End for

         ` All previously loaded (and locked) record were automatically released by loading
         `  another record
         ` The last processed record has to by unloaded 'manually'
      $errCode := OP Unload record (vConnectID;vTableCustomers)

      $ErrCode:=OP Delete bind ($BindInvID)
      $ErrCode:=OP Delete bind ($BindCustID)
```

See Also

OP Goto selected record, OP Load record, OP Set access mode, OP Unload record, SAVE RECORD.

OP New record

Records

version 1.5

OP New record (connectionID; bindID) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
bindID	Longint	→	Bind list ID
Function result	Longint	←	Error code result for the function

Description

OP New record adds a record in the file specified by bindID. The new record will contain the values specified for the fields defined in bindID. All other fields in the same table will be initialized to the default value.

To add a new record, you must first create and define the bind, assign the values to be used in the new record, and then call OP New record. The values to be given to the fields can also be defined before creating and defining the bind.

The new record is automatically unloaded after it has been added.

Error Codes

If OP New record executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
-9967	The record was not modified because it could not be loaded.
-9969	Invalid field type requested.
-9971	Field number is out of range.
-9972	Table number is out of range.
-9998	Duplicate index key .
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10137	The bind list does not exist.
10138	The bind list is not related to this file.
10139	The bind list is not defined.
10153	Unable to convert this type of data.
10154	This command cannot be executed right now.

Example

This example lets you copy customer records in the local database to the remote database. Assume that the same [Customers] table structure exists in both the local and the remote database.

```
C_LONGINT ($custBindID)
C_LONGINT (remTableID;remFieldID)
C_LONGINT ($errCode;vRecords;$unused)
   ` get local table name (in case it changes) and number
$CustTableName := Table name (->[Customers])
$CustTable := Table (->[Customers])
$fieldName := $CustTableName + Field name ($CustTable;1)

   ` Get the remote table ID
$errCode := OP Get one field number (vConnectID;$fieldName;remTableID;$unused)

   ` Create the bind list. It will be made of binds on the local database fields.
$errCode := OP Create bind ($custBindID)
For ($i;1; Count fields($CustTable))
   ` Get the local field name in the [Table]Field format, it MUST have the same name as
   ` the remote field
   $fieldName :=$CustTableName+ Field name ($custTable;$i)
   ` Bind using a pointer to the local field
   $errCode := OP Define bind by pointer ($custBindID;remTableID;$i;
                                                          Field ($CustTable;$i))
End for
ALL RECORDS ([Customers])
   ` Get a pointer to the field holding the customer ID to lookup
$custID:= ->[Customers]Customer ID
   ` Browse all the records in the local table
While ( Not ( End selection ([Customers])))
      ` Lookup the remote table to see if the record exists
   $errCode := OP Single query (vConnectID;remTableID;1;"=";$custID;vRecords)
   If (vRecords = 0)
      ` If the remote record does not exist : create it. The values to be used in the new
      ` record are in the local current record as defined in the bind list
⇒     $errCode := OP New record (vConnectID;$custBindID)
   Else
      ` The code here could take care of updating the remote record with the current
      ` local values.
   End if
   NEXT RECORD ([Customers])
End while
$errCode := OP Delete bind ($custBindID)
```

See Also

OP Load record, OP Set access mode, OP Update record, SAVE RECORD.

OP Sequence number

Records

version 1.5

OP Sequence number (connectionID; tableID; sequenceNumber) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
sequenceNumber	Longint	←	New sequence value for target table
Function result	Longint	←	Error code result for the function

Description

OP Sequence number returns the next available sequence number in sequenceNumber for tableID.

Error Codes

If OP Sequence number executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

See Also

OP New record, OP Update record, Sequence number.

OP Delete record

Records

version 1.5

OP Delete record (connectionID; tableID) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
Function result	Longint	←	Error code result for the function

Description

OP Delete record deletes the current record from tableID.

If there is no current record in tableID or if tableID is in read-only mode, OP Delete record has no effect.

If the record is locked, the record is not deleted and OP Delete record does not return an error. You must check the isLocked parameter of the OP Load record, OP Goto record, or OP Goto selected record function.

Error Codes

If OP Delete record executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9972	Table number is out of range.
-9986	Record locked during an automatic deletion action.
-9987	There are records related to this record .
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

This example lets you delete customer records in the remote database if they do not exist in the local database. Assume that the same [Customers] table structure exists in both the local and the remote database.

```
C_LONGINT (remTableID;remFieldID)
C_LONGINT ($errCode;vRecords;$unused;$lock)
C_LONGINT ($CustBind;$custLockBind)
C_STRING (10;remCustID)
ARRAY STRING (10;locCustIDs;0)
ARRAY STRING (10;remCustIDs;0)
ARRAY LONGINT (remCustRecordID;0)
```

```
   ` Get local table name (in case it changes) and number
$CustTableName := Table name (->[Customers])
$CustTable := Table (->[Customers])
$fieldName := $CustTableName + Field name (->[Customers]Customer ID)
   ` Get the remote table ID
$errCode := OP Get one field number (vConnectID;$fieldName;remTableID;remFieldID)
   ` Create a bind list to load remote customer IDs
$errCode:= OP Create bind ($custBind)
$errCode:=OP Define bind by pointer ($custBind;remTableID;remFieldID;->remCustIDs)
   ` Create another bind list to test record locks
$errCode := OP Create bind ($custLockBind)
$errCode := OP Define bind by pointer ($custLockBind;remTableID;
                                                    remFieldID;->remCustID)
ALL RECORDS ([Customers])
   ` Load the local customer IDs in an array
Selection to array ([Customers];[Customers]Customer ID;locCustIDs)

$errCode := OP All records (vConnectID;remTableID)
   ` Load the remote customer IDs in an array
$errCode := OP Selection to array (vConnectID;$custBind)
   ` Load also the record numbers for faster retrieval later
$errCode := OP Get record numbers (vConnectID;remTableID;remCustRecordID)
$deleteCount := 0
   ` For each entry in the remote customer IDs
For ($i;1; Size of array (remCustIDs))
      ` Look up the remote customer ID into the local customer IDs array
   $local := Find in array (locCustIDs;remCustIDs {$i})
   If ($local = -1)
         ` If it was not found, the delete it from the remote table. First make it the
         ` current record, load the ID and lock it, by going to its direct record number
      $errCode := OP Goto record (vConnectID;remTableID;remCustRecordID{$i};
                                                    $custLockBind;$lock)
      ` We want to make sure that the record was not deleted and replaced by another
      ` one so we check that the Customer ID is really the one we are looking for.
      If (remCustID = locCustIDs {$i})
         `we'll assume the record is not already locked by another process (i.e. $lock=0)
⇒        $errCode := OP Delete record (vConnectID;remTableID)
         $deleteCount := $deleteCount +1
      End if
   End if
End for
ALERT ("There was "+ String ($deleteCount) + " records deleted on the remote table")
$errCode := OP Delete bind ($custBind)
$errCode := OP Delete bind ($custLockBind)
```

See Also

DELETE RECORD, OP Delete selection, OP Single query, OP Update record.

OP Goto record

Records

version 1.5

OP Goto record (connectionID; tableID; recordNumber; bindID; lockStatus) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
recordNumber	Longint	→	Absolute record number in the table
bindID	Longint	→	Bind list ID
lockStatus	Longint	←	Lock status = 1 means the record is locked
Function result	Longint	←	Error code result for the function

Description

OP Goto record makes recordNumber the current and only record in the selection.
If you pass 0 in bindID or omit the last two parameters, OP Goto record changes only the current record pointer. If you pass a valid bind list ID, the routine will also download the fields accordingly. In this case, if the access mode to the table is read/write, the routine will additionally try to give you access to the record in read/write mode. If this last operation is successful, the routine returns 0 in lockStatus. If the record is already in use by another process, the routine returns 1 in lockStatus.
By testing the lockStatus parameter, you know whether or not you can subsequently modify or delete the record using OP Update record or OP Delete record.
If you obtained the current record in read/write mode, you must release the record for the other processes by using OP Unload record once you are done with the record (unless you deleted it).

Error Codes

If OP Goto record executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-9972	Table number is out of range.
-10003	Record is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

See example for OP Delete record.

See Also

GOTO RECORD, OP Delete record, OP Current Record Number, OP Get record numbers, OP Goto selected record, OP Load record, OP Single query.

OP Get record numbers

Records

version 1.5

OP Get record numbers (connectionID; tableID; recordNumbers{; firstRecordNumber{; lastRecordNumber}}) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
recordNumbers	Array	←	Absolute record numbers in target table
firstRecordNumber	Longint	→	Relative number of first record in the current selection
lastRecordNumber	Longint	→	Relative number of last record in the current selection
Function result	Longint	←	Error code result for the function

Description

OP Get record numbers returns in recordNumbers the record numbers for the selection of records in tableID. recordNumbers is an array of type Longint or Real.

You can enter values in the optional firstRecordNumber and lastRecordNumber parameters to limit the number of records whose numbers are returned in recordNumbers. If firstRecordNumber = 0, OP Get record numbers returns the record numbers of all selected records. Remember that firstRecordNumber and lastRecordNumber are selected record numbers, but the numbers returned in the recordNumbers array are record numbers.

Error Codes

If OP Get record numbers executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10135	Invalid parameter type.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

See example for OP Delete record.

See Also

OP Delete record, OP Current Record Number, OP Selection to array, OP Single query, SELECTION TO ARRAY.

OP Current Record Number

Records

version 1.5

OP Current Record Number (connectionID; tableID; recordNumber; recordsInSelection) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
tableID	Longint	→	Number of the table in the database
recordNumber	Longint	←	Number in selection of the current record
recordsInSelection	Longint	←	Number of records in current selection
Function result	Longint	←	Error code result for the function

Description

OP Current Record Number returns in recordNumber the number of the current record. It returns in recordsInSelection the number of records in the current selection of tableID.

Error Codes

If OP Current Record Number executes successfully, it retursn 0. Otherwise, this function returns one of the following errors:

Error	Code Description
-9956	4D Open version not compatible with 4D Server.
-32001	A query in asynchronous mode is already in progress.
-9972	Table number is out of range.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

See Also

OP Get record numbers.

14

Utilities

The following routines are described in this section:

• OP Get error text - gets a description of an error that you received after calling a 4D Open for 4th Dimension function. When you pass the error number, this routine returns the error message associated with that error number.

• OP Set option - sets a variety of options for 4D Open for 4th Dimension.

• OP Get option - finds the value of an option that has been set by OP Set option.

• OP Flush Buffers - immediately saves the data buffers to disk.

• OP Get version number - returns the version number of 4D Open for 4D.

OP Set option (optionNumber; optionValue) → Longint

Parameter	Type		Description
optionNumber	Longint	→	Number of the option to set
optionValue	Longint	→	Value to which to set the option
Function result	Longint	←	Error code result for the function

Description

OP Set option lets you set some operating options for 4D Open for 4th Dimension.

The codes for optionNumber and optionValue are listed in the following table:

Option	Value	Description
1	1 / 0	Turn parameter checking mode On/Off. On is the default.
2	1 / 0	Turn Error dialog box On/ Off. On is the default.
3	1 / 0	Caching structure information is or is not authorized.
4	n	Element bound when arrays are used with record routines.Default is element zero.
(NCID*65536)+ Code table.	1 / 0	Set a network component option. See the following code

Option 1 - Parameter checking mode

When parameter checking is on, 4D Open for 4th Dimension checks the validity of the parameters. For example, if a parameter is the table number of the served database, the program senda a request to check whether or not the table exists. This option is on by default. You can turn it off by calling OP Set option(1;0). Checking the parameters slows down operations.

Option 2 - Error dialog box option

This option, which is on by default, displays the Error dialog box when an error occurs.

Option 3 - Caching structure information

This option, which is off by default, tells 4D Open for 4th Dimension whether or not caching structure calls can be made. You can turn it on by calling OP Set option(3;1). Once the option has been turned on, you can use the OP Cache structure routine. Caching the structure speeds up some operations.

Option 4 - Binding with arrays

When you use arrays in binds, the element 0 is used for record routines by default. You may change this element using OP Set option (4;ElementNumber).

Network component options

OP Set option allows you to set some options for your network component. This features allows expert users to change some of the connection parameters. 4D Open for 4th Dimension does not check the options you have set; some changes may lead to problems. When used this way,

optionValue = (NCID * 65536) + Code of option to set

You can obtain NCID (the network component ID) using OP Get network component info.

Use OP Set option to set a network component option only when the network component is not currently initialized.

The codes for the options to set require detailed knowledge of the network components and should be used only by expert programmers. The codes are:

Code	Name	Description
1	Version	This field indicates the version of the network component. You must NOT modify the value of this field.
2	LockDelay	This field is used by the ADSP component when used with 4D Client. You must NOT modify the value of this field.
3	IdleDelay	This field must be equal to zero.
4	BufferSize	This field indicates the size (in bytes) of the communication buffers used by the network component.
5	PackOption	This field tells the network component to perform automatic data compression when sending a data buffer over the network.
6	MinPackSize	If PackOption is equal to 1 (data is compressed), this field specifies the minimum size (in bytes) of the buffers to be compressed.
7	MaxPackSize	If PackOption is equal to 1 (data is compressed), this field specifies the maximum size (in bytes) of the buffers to be compressed.
8	DelayVoyant	This field must be equal to 0.
9	Product ID	This field specifies the type of application in which the network component is running.
10	Flags	This field is a set of 32 bits. Each bit is set to either 1 or 0 depending on the value of the corresponding option for the network component.

Error Codes

If OP Set option executed successfully, it returns 0. Otherwise, it returns the following errors:

Error Code	Description
-192	Resource not found.
10128	The 4D Open for 4th Dimension package has not been initialized.
10132	The network component is currently in use.
10148	Unknown option requested to 4D Open for 4th Dimension.
10154	This command cannot be executed right now.
10165	Unknown network component option.

Example

In this example, 4D Open for 4D checks the validity of parameters while in a debugging phase. At a more advanced stage of debugging, it also turns structure caching ON.

```
If (<>vDebugLevel >= checkParams )
   ` Debug phase is in progress, set parameter checking on for improved security
   ` Set parameters checking option(option 1)  ON (value 1) for 4D Open
   $errCode := OP Set option (1;1 )

Else
   ` Debug phase is very advanced, remove parameter checking for improved
                                                                    performance
   ` Set parameters checking option(option 1)  OFF (value 0) for 4D Open
   $errCode := OP Set option (1;0 )
   If (<>vDebugLevel = final )
      ` No more debugging, server structure is final, start caching structure information
      $errCode : = OP Set option (3;1)
      $errCode := OP Cache structure (vConnectID;1)
   Else
      $errCode : = OP Set option (3;0)
   End if
End if
```

See Also

OP Get option.

OP Get option

Utilities

version 1.5

OP Get option (optionNumber; optionValue) → Longint

Parameter	Type		Description
optionNumber	Longint	→	Number of the option to get
optionValue	Longint	←	Current setting of the option
Function result	Longint	←	Error code result for the function

Description

OP Get option gets the current values for the options that you can set with OP Set option. You can use OP Get option at any time.

The codes for optionNumber and optionValue are listed in the table for OP Set option.

Error Codes

If OP Get option executes successfully, it return 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-192	Resource not found.
10128	The 4D Open for 4th Dimension package has not been initialized.
10132	The network component is currently in use.
10148	Unknown option requested to 4D Open for 4th Dimension.
10154	This command cannot be executed right now.
10165	Unknown network component option.

See Also

OP Set option.

OP Flush Buffers (connectionID) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
Function result	Longint	←	Error code result for the function

Description

OP Flush Buffers immediately saves the data buffers to disk. All changes that have been made to the database are stored on disk.

Error Codes

If OP Flush Buffers executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error	Code Description
-32001	A query in asynchronous mode is already in progress.
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

OP Get version number

Utilities

version 6.0

OP Get version number → Version number

Parameter	Type		Description
This command does not require any parameters			
Function result	Version number	←	Version number of 4D Open for 4D

Description

OP Get version number gives you the version number of 4D Open for 4D.

This command returns the version number (6xx) in the High Word and the build number in the Low Word.

Examples

1. Version 5:

```
⇒   $Version:=OP Get Version Number
       ` In order to get the High Word
    StringVers:=String($Version\ 65536)
       ` In order to get the Low Word
    StringVers:=StringVers+" Build: "+String($Version% 65536)
```

2. Version 6:

```
⇒   $Version:=OP Get Version Number
       ` In order to get the High Word
    StringVers:=String($Version >> 16)
       ` In order to get the Low Word
    StringVers:=StringVers+" Build: "+String($Version & 65535)
```

15

Users and Groups

The following routines are described in this section:

• OP Get user list - gets the user list for either the Designer or the Administrator of the database.

• OP Enter password - to use the 4th Dimension password dialog boxes and then retrieve the selected user name as well as the entered password.

• OP Get users and groups - gets detailed information about the users and groups defined for a database.

OP Get user list (connectionID; listSelect; userNames; userConnections; userLastLogin) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
listSelect	Longint	→	Selects the Administrator (1) or Designer (0) user list
userNames	Array	←	Array of user names
userConnections	Array	←	Array of number of connections per user
userLastLogin	Array	←	Array of 'last user login' dates
Function result	Longint	←	Error code result for the function

Description

OP Get user list retrieves the user list for either the Designer or the Administrator of the database. It also retrieves the number of connections for each user and the date of each user's last connection.

There are two sets of users in a 4th Dimension structure file: the Designer and Administrator maintain their own sets of users. listSelect determines which set of users is retrieved by this command.

UserNames can be an array of type String or Text.

UserConnections can be an array of type String, Text, Real, Integer, or Long Integer.

UserLastLogins can be an array of type String, Text, or Date.

Example

This example uses the OP Get user list function to find login information about the users.

```
ARRAY TEXT (arUserNames;0)
ARRAY DATE (arLastLog;0)
ARRAY INTEGER (arUserConns;0)
C_LONGINT ($errCode)
C_TEXT ($message)

   ` Gather infor about Administrator-created accounts
⇒  $errCode := OP Get user list (vConnectID;1;arUserNames;arUserConns;arLastLog)

$message:=""

For ($i;1;Size of array (arUserNames))
   If (asLastLog {$i} < Current date -7)
   ` If no login for that user has occured
      ` Alert Administrator
      $message := $message + arUserNames {$i} + ", "
   End if
End for

$message := $message + Char (Carriage return) + " have not logged in for a week"
$message := $message + Char (Carriage return) + " you should disable their account
                                                          until their return"

SendMessage ($mess;"admin_4D@acme.com")
```

See Also

OP Get users and groups.

OP Enter password (userNames; userName; userPassword) → Longint

Parameter	Type		Description
userNames	Array or String	→	List of user names (or empty string)
userName	String	←	Selected (or entered) user name
userPassword	String	←	Entered password
Function result	Longint	←	Error code result for the function

Description

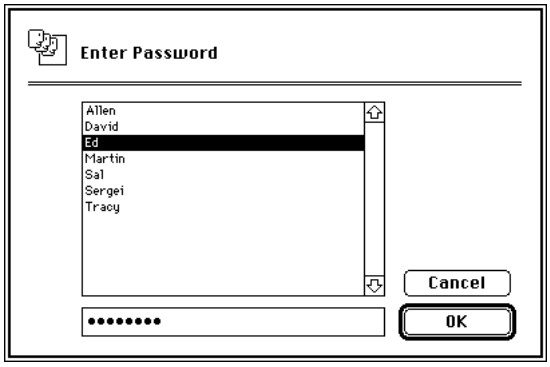
OP Enter password allows you to display the 4th Dimension password dialog boxes so that users can select or enter a name as well as a password.

This routine only displays the 4th Dimension password dialog box. It does not check the users or their passwords in the databases. To do so, use the OP Open connection routine.

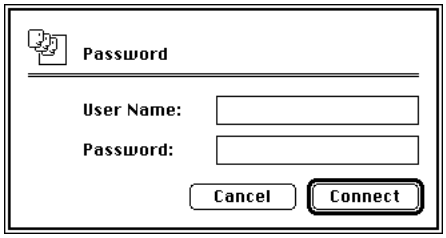
userName is the user name that was either entered or selected depending on the dialog box displayed. In 4th Dimension, the maximum length for a user name is 30 characters.

userPassword is the password entered by the user. In 4th Dimension, the maximum length for a password is 15 characters.

If userNames is a non-empty array of type Alpha or Text, the following dialog box appears:



If userNames is an empty array, the following dialog box appears:



Error Codes

If the user clicked the OK or Connect button, OP Enter password returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
10147	The user clicked Cancel in the Enter password dialog box.
10154	This command cannot be executed right now.

Example

This example stores the user name and password in the application preference table. It allows for automatic login at any time.

```
   ` Method GetUserPasswd
C_LONGINT ($errcode)
ARRAY TEXT (arUserNames;0)
ARRAY DATE (arLastLog;0)
ARRAY INTEGER (arUserConns;0)
C_STRING (30;vUserName)
C_STRING (15;vUserPasswd)

   ` Get the list of user names from the remote server
$errCode := OP Get user list (vConnectID;arUserNames;arLastLog;arUserConns)

   ` Get the user to select his remote name and enter his password
⇒  $errCode := OP Enter password (arUserNames;vUserName;vUserPasswd)

READ WRITE ([Preferences])
   `Look for an existing UserName preference
QUERY ([Preferences];[Preferences]Token = "RemoteLoginName")
If (Records in selection ([Preferences])=0)
   NEW RECORD ([Preferences])
End if
[Preferences]Token := "RemoteLoginName"
[Preferences]Value := vUserName
SAVE RECORD ([Preferences])
   `Look for an existing UserPassword preference
QUERY ([Preferences];[Preferences]Token = "RemoteLoginPasswd")
```

```
If (Records in selection ([Preferences])=0)
   NEW RECORD ([Preferences])
End if
[Preferences]Token := "RemoteLoginPasswd"
[Preferences]Value := vUserPasswd
SAVE RECORD ([Preferences])
UNLOAD RECORD ([Preferences])

ARRAY TEXT (arUserNames;0)
ARRAY DATE (arLastLog;0)
ARRAY INTEGER (arUserConns;0)

   ` Method OpenRemoteConnection -> connectionID
C_LONGINT (netCompID;$serverID;$connectID)
C_LONGINT ($0)
READ ONLY ([Preferences])
QUERY ([Preferences];[Preferences]Token = "RemoteServerAdress")
$address := [Preferences]Value
QUERY ([Preferences];[Preferences]Token = "RemoteServerProtocol")
$protocol := [Preferences]Value
QUERY ([Preferences];[Preferences]Token = "RemoteLoginName")
$userName := [Preferences]Value
QUERY ([Preferences];[Preferences]Token = "RemoteLoginPasswd")
$userPasswd := [Preferences]Value

If ( ($address="") | ($protocol="") | ($userName="") | ($userPasswd="") )
      `Return 'prefs not availbale' error code
   $0 := -1
Else
   $errCode := LoadNetComp ($protocol;->netCompID)
   If ($errCode = 0)
         ` Turn the adress string into a server entry
      $errCode := OP Find 4D Server (netCompID;$address;serverID)
   End if
   If ($errCode =0)
      ` Open connection with the server
      $errCode := OP Open connection ($servID; $connID; "CustService"; $userName;
                                                          $userPasswd;"OrderEntry")
   End if
   If ($errCode = 0)
      $0 :=$connID
   Else
      $0 := $errCode
   End if
End if
```

See Also

OP Open connection.

OP Get users and groups

Users and Groups

version 1.5

OP Get users and groups (connectionID; listSelect; userNames; userConnections; userStartMethod; userLastLogins; ownerList; groupNames; groupSizes; groupOwners; members) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
listSelect	Longint	→	Selects the Administrator (1) or Designer (0) user list
userNames	Array	←	Array of user names
userConnections	Array	←	Array of number of connections per user
userStartMethod	Array	←	Array of user startup methods
userLastLogins	Array	←	Array of 'last user login' dates
ownerList	Array	←	Array of object owners
groupNames	Array	←	Array of group names
groupSizes	Array	←	Array of number of users per group
groupOwners	Array	←	Array of group owners
members	Array	←	2D array of user IDs per group
Function result	Longint	←	Error code result for the function

Description

OP Get users and groups returns information about the users and groups defined for a database.

You can obtain this information only if you are connected as the Designer or the Administrator.

There are two sets of users in a 4th Dimension structure file. The Designer and Administrator maintain their own set of users. listSelect determines which set of users is retrieved by this command.

userNames, userStartMethod, and groupNames can be arrays of type Text or String.

userConnections, ownerList, groupSizes, and groupOwners can be arrays of type String, Text, Real, Integer, or Long Integer.

userLastLogins can be an array of type String, Text, or Date.

members is a two-dimensional array of type String, Text, Integer, or Long Integer. Each element in the array represents a group and is an array of user or group IDs in that group.

The values returned in the second dimension are numeric and can be positive or negative depending on the nature of the members.

The following table shows how to interpret the values returned in the Members array:

Value	Designates	Description
from 1 to 15000	Users in Designer set	Gives the element number of the user name in the UserName array returned by a call with WhichSet = 0. [ElementNumber = Value]
above 15000	Groups in Designer set	Gives the element number plus 15000 of the group name in the GroupName array returned by a call with WhichSet = 0. [ElementNumber = Value - 15000]
from -11 to -15000	Users in Administrator set	Gives the negative element number minus 10 of the user name in the UserName array returned by a call with WhichSet = 1. [Element number = ABS(Value)-10]
below -15000	Groups in Administrator set	Gives the negative element number minus 15000 of the group name in the GroupName array returned by a call with WhichSet = 1. [Element number = ABS(Value)-15000]

See Also

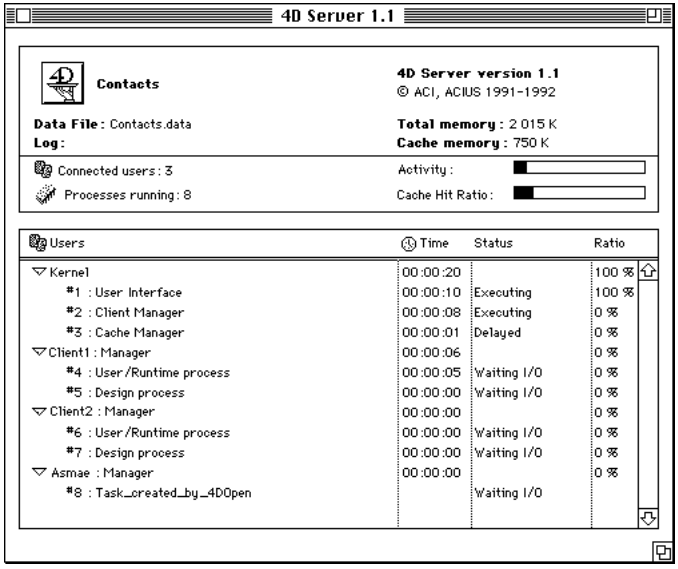
OP Get User List.

16

Server Information

The following routines are described in this section:

- OP Get server date - retrieves the date on the server.
- OP Get server time - retrieves the time on the server.
- OP Get server version - gets the version, release and update numbers of 4D Server.
- OP Count connected users - gets the number of users currently connected to the same 4D Server database.
- OP Count user processes - gets the number of user processes.
- OP Get process list - gets information about the processes running on the server machine.



OP Get server date

Server Information

version 1.5

OP Get server date (connectionID; serverDate) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
serverDate	Date	←	Current system date on server
Function result	Longint	←	Error code result for the function

Description

OP Get server date returns in serverDate the system's date on the server indicated by connectionID.

Error Codes

If OP Get server date executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

This example uses the server's system date and time to timestamp order entry. The 'getTimeStamp' method ensures that even if some of the worstations have a wrong time or date, they will get the proper settings from the server.

```
      ` GetTimeStamp (connectionID) -> Timestamp
      ` ConnectionID     number    -> connection ID with targer server
      ` Timestamp     String          -> YYYYMMDDHHMMSS
   C_LONGINT ($1;$connID)
   C_STRING (14;$ts;$0)
   C_LONGINT ($errCode)
   C_DATE ($serverDate)
   C_TIME ($serverTime)

   $ConnId := $1
⇒  $errCode := OP Get server date ($connID;$serverDate)
   $errCode := OP Get server time ($connID;$serverTime)

   $ts := String ( Year of ($serverDate); "####")
   $ts := $ts + String ( Month of ($serverDate); "##")
   $ts := $ts + String ( Day of ($serverDate); "##")
   $ts := $ts + Substring ( Time string ($serverTime); 1 ; 2 )
   $ts := $ts + Substring ( Time string ($serverTime); 4 ; 2 )
   $ts := $ts + Substring ( Time string ($serverTime); 7 ; 2 )
   $0 := $ts
```

See Also

Current date, Current time, OP Get server time.

OP Get server time

Server Information

version 1.5

OP Get server time (connectionID; serverTime) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
serverTime	Longint	←	Current system time on server
Function result	Longint	←	Error code result for the function

Description

OP Get server time returns in serverTime the system's time on the server indicated by connectionID.

Error Codes

If OP Get server time executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

Example

See example for OP Get server date.

See Also

Current time, OP Get server date.

OP Count connected users

Server Information

version 1.5

OP Count connected users (connectionID; countUsers) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
countUsers	Longint	←	Number of users currently connected
Function result	Longint	←	Error code result for the function

Description

OP Count connected users returns in countUsers the number of users currently connected to the same 4D Server designated by connectionID.

Error Codes

If OP Count connected users is executed successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

See Also

Count users.

OP Count user processes (connectionID; countProcesses) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
countProcesses	Longint	←	Number of running users processes
Function result	Longint	←	Error code result for the function

Description

OP Count user processes returns in countProcesses the number of user processes on the designated server.

Error Codes

If OP Count user processes executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

See Also

Count user processes, OP Count connected users.

OP Get server version

Server Information

version 1.5

OP Get server version (connectionID; serverVersion; serverRelease; serverUpdate) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
serverVersion	Longint	←	Target server version number
serverRelease	Longint	←	Target server release number
serverUpdate	Longint	←	Target server update number
Function result	Longint	←	Error code result for the function

Description

OP Get server version returns the version, release and update numbers of 4D Server. For example, 4D Server 1.2 would be returned as: serverVersion = 1, serverRelease = 2, and serverUpdate = 0.

Error Codes

If OP Get server version executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
10128	The 4D Open for 4th Dimension package has not been initialized.
10136	The connection does not exist.
10154	This command cannot be executed right now.

OP Get process list

Server Information

version 1.5

OP Get process list (connectionID; countProcesses; userNames; stationNames; taskNames; timeSpent; taskStatus) → Longint

Parameter	Type		Description
connectionID	Longint	→	Connection ID with target server
countProcesses	Longint	←	Number of running processes on server
userNames	Array	←	Array of processes' user names
stationNames	Array	←	Array of processes' station names
taskNames	Array	←	Array of processes names
timeSpent	Array	←	Array of processes' elapsed time
taskStatus	Array	←	Array of processes' statuses
Function result	Longint	←	Error code result for the function

Description

OP Get process list gets information about the processes running on the server machine.
userNames, stationNames, and taskNames are String or Text arrays.
timeSpent and taskStatus are String, Text, Integer, Long Integer, or Real arrays.
The values returned in the taskStatus array depend on the type of array passed, based on the following table:

Numeric	Alpha
0	Executing
1	Delayed
2	Waiting for user event
3	Waiting for Input/Output
4	Waiting for internal flag
5	Paused
6	Hidden modal dialog

Error Codes

If OP Get process list executes successfully, it returns 0. Otherwise, this function returns one of the following errors:

Error Code	Description
-108	Not enough memory to perform this operation.
10128	The 4D Open for 4th Dimension package has not been initialized.
10135	Invalid parameter type.
10136	The connection does not exist.
10154	This command cannot be executed right now.

See Also

PROCESS PROPERTIES, OP Count user processes.

17

Processes

OP Process number

Processes

version 6.0.6

OP Process number (connectID; processName; processID) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
processName	String	→	Name of the process
processID	Integer	←	ID of the process
Function result	Longint	←	Error code

Description

The command OP Process number returns the process number (processID) of the process specified by processName.

See Also

OP Execute On Server.

OP Set Process Variable

Processes

version 6.0.6

OP Set Process Variable (connectID; processID; procVar; locVar) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
processID	Integer	→	ID of the process
procVar	String	→	Name of process variable to write
locVar	String	→	Name of local variable to read value from
Function result	Longint	←	Error code

Description

The command OP Set Process Variable assigns the value of the process variable locVar to the process variable procVar, specified by processID.

See Also

OP Get Process Variable.

OP Get Process Variable

Processes

version 6.0.6

OP Get Process Variable (connectID; processID; procVar; locVar) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
processID	Integer	→	ID of the process
procVar	String	→	Name of process variable to read
locVar	String	→	Name of local variable to write value to
Function result	Longint	←	Error code

Description

The command OP Get Process Variable assigns the value of the process variable procVar, specified by processID, to the local variable locVar.

See Also

OP Set Process Variable.

OP Execute On Server

Processes

version 6.0.6

OP Execute On Server (connectID; methodName; stackSize; processName; processID{; param}{; param2; ...; paramN}) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
methodName	String	→	Name of the process method
stackSize	Longint	→	Stack size of the process
processName	String	→	Name of the process
processID	Integer	←	ID of the process
param	String	→	Parameter(s) to the method (≤10)
Function result	Longint	←	Error code

Description

The command OP Execute On Server starts a new process on the server machine. It has a limit of 10 optional parameters.

Note: This function may return a processID equal to 0 (zero).

OP Execute On Server does not return any error code if the method does not exist on the server — this case means that the process hasn't been started properly and should be handled as an error.

Note that you must pass variable names as strings in the param parameter, and not the values themselves. See the following example:

```
Param1:="Smith"
Param2:="John"
Param3:="M"
⇒ $error:=OP Execute On Server($connID;"myMeth";1024*64;$procID;"Param1";
                                                    "Param2";"Param3")
```

See Also

OP Process number.

OP Set Semaphore

Processes

version 6.0.6

OP Set Semaphore (connectID; semaphoreName; semaphoreState) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
semaphoreName	String	→	Name of the semaphore
semaphoreState	Longint	←	State of the semaphore
Function result	Longint	←	Error code

Description

The command OP Set Semaphore sets semaphoreName and returns its current state in semaphoreState (1=semaphoreName is set, else 0).

See Also

OP Check Semaphore, OP Clear Semaphore.

OP Clear Semaphore

Processes

version 6.0.6

OP Clear Semaphore (connectID; semaphoreName) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
semaphoreName	String	→	Name of the semaphore
Function result	Longint	←	Error code

Description

The command OP Clear Semaphore erases semaphoreName.

See Also

OP Check Semaphore, OP Set Semaphore.

OP Check Semaphore

Processes

version 6.0.6

OP Check Semaphore (connectID; semaphoreName; semaphoreState) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
semaphoreName	String	→	Name of the semaphore
semaphoreState	Longint	←	State of the semaphore
Function result	Longint	←	Error Code

Description

The command OP Check Semaphore returns in semaphoreState the current status of the existing semaphore semaphoreName.

See Also

OP Clear Semaphore, OP Set Semaphore.

18

Sets

OP Create Empty Set

Sets

version 6.0.6

OP Create Empty Set (connectID; tableID; setName) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
tableID	Integer	→	ID of the table
setName	String	→	Name of the empty set to create
Function result	Longint	←	Error code

Description

The command OP Create Empty Set creates a new empty set for the specified table.

See Also

OP Create Set.

OP Create Set

Sets

version 6.0.6

OP Create Set (connectID; tableID; setName) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
tableID	Integer	→	ID of the table
setName	String	→	Name of the set to create
Function result	Longint	←	Error code

Description

The command OP Create Set creates a new set for specified table and places the current selection in the specified set.

See Also

OP Create Empty Set.

OP Use Set

Sets

version 6.0.6

OP Use Set (connectID; setName) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
setName	String	→	Name of the set to use
Function result	Longint	←	Error code

Description

The command OP Use Set makes the records in the specified set the current selection for the table to which the set belongs.

See Also

OP Clear Set.

OP Add To Set

Sets

version 6.0.6

OP Add To Set (connectID; tableID; setName) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
tableID	Integer	→	ID of the table
setName	String	→	Name of the set
Function result	Longint	←	Error code

Description

The command OP Add To Set adds the current record of the specified table to the specified set.

Example

See the example of the function routine OP Remove From Set.

See Also

OP Is In Set, OP Remove From Set.

OP Remove From Set

Sets

version 6.0.6

OP Remove From Set (connectID; tableID; setName) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
tableID	Integer	→	ID of the table
setName	String	→	Name of the set
Function result	Longint	←	Error code

Description

The command OP Remove From Set removes from setName the current record of the specified table.

Example

Here is a typical use of the Sets comands:

```
$ErrCode:=OP Single query (vConnectID;$Server_Table;$Server_Field;"=";>vValue;
                                                                    vRecords)
$ErrCode:=OP Create Set (vConnectID;$Server_Table;"SetA")
$ErrCode:=OP All records (vConnectID;$Server_Table)
$ErrCode:=OP Create Set (vConnectID;$Server_Table;"SetNoA")
ARRAY LONGINT(vRecNums;0)
$ErrCode:=OP All records (vConnectID;$Server_Table)
$ErrCode:=OP Get record numbers (vConnectID;$Server_Table;vRecNums)
For ($k;1;Size of array(vRecNums))
   $ErrCode:=OP Goto record (vConnectID;1;vRecNums{$k})
   $ErrCode:=OP Is In Set (vConnectID;"SetA";vRecNums{$k};vBool)
   If (vBool=1)
⇒     $ErrCode:=OP Remove From Set (vConnectID;$Server_Table;"SetNoA")
   End if
End for
```

See Also

OP Add To Set, OP Is In Set.

OP Clear Set

Sets

version 6.0.6

OP Clear Set (connectID; setName) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
setName	String	→	Name of the set to clear
Function result	Longint	←	Error code

Description

The command OP Clear Set clears set from memory and frees the memory used by set.

See Also

OP Remove From Set.

OP Is In Set

Sets

version 6.0.6

OP Is In Set (connectID; setName; recordNumber; isInside) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
setName	String	→	Name of the set
recordNumber	Longint	→	Absolute number of the record
isInside	Longint	←	Is the record in the set ? 1 = yes, 0 = no
Function result	Longint	←	Error code

Description

The command OP Is In Set tests whether or not the record recordNumber of the setName's table is in setName and sets isInside to 0 if the record is not in it.

OP Is In Set requires an absolute record number, returned by the OP Get records numbers function.

See Also

OP Get record numbers, OP Records In Set, OP Remove From Set.

OP Records In Set

Sets

version 6.0.6

OP Records In Set (connectID; setName; totalRecords) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
setName	String	→	Name of the set to clear
totalRecords	Longint	←	Number of the records in the set
Function result	Longint	←	Error code

Description

The command OP Records In Set returns in totalRecords the number of records present in the specified set.

See Also

OP Is In Set, OP Remove From Set.

OP Copy Set

Sets

version 6.0.6

OP Copy Set (connectID; sourceSet; destinationSet) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
sourceSet	String	→	Source set name
destinationSet	String	→	Destination set name
Function result	Longint	←	Error code

Description

The command OP Copy Set copies the sourceSet set into the destinationSet set.

See Also

OP Difference Set, OP Intersection Set, OP Union Set.

OP Union Set

Sets

version 6.0.6

OP Union Set (connectID; sourceSet1; sourceSet2; destSet) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
sourceSet1	String	→	First source set name
sourceSet2	String	→	Second source set name
destSet	String	→	Destination set name
Function result	Longint	←	Error code

Description

The command OP Union Set creates a set that contains all the records from the sets sourceSet1 and sourceSet2.

See Also

OP Difference Set, OP Intersection Set.

OP Intersection Set

Sets

version 6.0.6

OP Intersection Set (connectID; sourceSet1; sourceSet2; destSet) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
sourceSet1	String	→	First source set name
sourceSet2	String	→	Second source set name
destSet	String	→	Destination set name
Function result	Longint	←	Error code

Description

The command OP Intersection Set compares sourceSet1 and sourceSet2, then selects only the records that are present in both and place them in destSet.

See Also

OP Difference Set, OP Union Set.

OP Difference Set

Sets

version 6.0.6

OP Difference Set (connectID; sourceSe1; sourceSet2; destSet) → Longint

Parameter	Type		Description
connectID	Longint	→	Unique ID for the established connection
sourceSe1	String	→	First source set name
sourceSet2	String	→	Second source set name
destSet	String	→	Destination set name
Function result	Longint	←	Error code

Description

The command OP Difference Set compares sourceSet1 and sourceSet2 and excludes all records that are present in sourceSet2 from the destSet.

See Also

OP Intersection Set, OP Union Set.

19

Error Codes

The following are the error codes concerning 4D Open for 4th Dimension.

-10132	Some connection parameters are invalid
-10131	The connection has been aborted
-10130	State of the connection does not allow you to continue this session
-1005	Incorrect value in Get/SetOption
-10050	Unknown option in Get/SetOption
-10021	No server was found while using OP Find 4D server
-10020	No server was selected while using OP Select 4D server
-10003	Bad connection parameters
-10002	The connection for this process has been disrupted
-10001	The actual connection to the database has been disrupted
2	The user clicked the Other button while using OP Select 4D Server
10128	The 4D Open for 4th Dimension package has not been initialized
10129	The network component was not found
10130	The initialization of the network component failed
10131	The network component has not been initialized
10132	The network component is currently in use. It cannot be deinitialized
10133	The server does not exist
10134	The server is currently in use. It cannot be deleted
10135	Invalid parameter type
10136	The connection does not exist
10137	The context does not exist
10138	The context is not related to this file
10139	The context is not defined
10140	The local field does not exist
10141	The local file does not exist
10142	You cannot bind this type of field
10143	Some resources are missing, the command can be executed
10144	The 4D Pointer is equal to NIL
10145	A 4D pointer was expected
10146	Invalid field name(s)
10147	The user clicked on Cancel in the password dialog
10148	Unknown option requested to 4D Open for 4th Dimension
10149	Invalid local file number
10150	Invalid local field number
10151	Invalid search operator
10152	Invalid logical operator
10153	Unable to convert this type of data
10154	This command cannot be executed right now
10155	Invalid sorting order
10156	Empty search or sort definition
10157	Duplicated bind entry

10158	The field is not an indexed alphanumeric field
10159	The array(s) contain or will contain too many elements
10160	Invalid password system set
10161	Array element out of range
10162	The array(s) contain no elements
10163	Bind entry not found
10164	Selected record number(s) incorrect or out of range
10165	Network component option not implemented

See Also

OP Get error text.

Command Index

A

C

D

E

F

G

I

L

M

N

O

P

R

S

U

V